Rigorous Numerical Approaches in Electronic Structure Theory

Australian
National
University

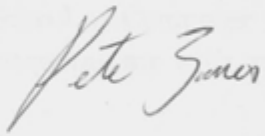
A thesis submitted for the degree of Doctor of Philosophy
of the Australian National University

Pete Peerapong Janes

September 2011

This document was produced using TEX , LATEX and BIBTEX

I declare that the work in this thesis is entirely my own and that to the best of my knowledge it does not contain any materials previously published or written by another person except where otherwise indicated.



Pete Peerapong Janes
23 September 2011

Acknowledgements

I am grateful to my supervisor Alistair Rendell for his support throughout the years. His guidance and advice were crucial in improving the focus of my research work at many critical moments. I would also like to thank the members of my supervisory panel: Stephen Roberts and Michael Collins for their advice during the important exploratory stage of this work.

I am appreciative of the help provided by Peter Strazdins in keeping Alcatraz and Mavericks in working order while I was running my interval analysis experiments. I want to to thank Carlile Lavor, Leo Liberti, and Nelson Maculan for their valuable advice during the initial part of my work on deterministic global optimization. I would like to thank Leo Liberti especially for hosting my visit to the Ecole Polytechnique at such short notice, which turned out to be very influential to my subsequent research work. I am indebted to Pietro Belotti for his assistance with implementation work in COUENNE.

I want to acknowledge Warren Armstrong, Josh Milthorpe, and Jie Cai for their valuable assistance in proofreading this thesis. I would like to thank Joseph Antony and Jin Wong for helpful suggestions on my research. I would also like to thank the administrative staff at the Research School of Computer Science, especially Julie Arnold and Deanne Drummond, for assisting me with my travel arrangements, and being so helpful overall.

Finally, I would like to thank my family and friends for their patience and support. I hope I can return my gratitude in the coming years.

Abstract

Electronic structure theory concerns the description of molecular properties according to the postulates of quantum mechanics. For practical purposes, this is realized entirely through numerical computation, the scope of which is constrained by computational costs that increases rapidly with the size of the system.

The significant progress made in this field over the past decades have been facilitated in part by the willingness of chemists to forego some mathematical rigour in exchange for greater efficiency. While such compromises allow large systems to be computed feasibly, there are lingering concerns over the impact that these compromises have on the quality of the results that are produced. This research is motivated by two key issues that contribute to this loss of quality, namely i) the numerical errors accumulated due to the use of finite precision arithmetic and the application of numerical approximations, and ii) the reliance on iterative methods that are not guaranteed to converge to the correct solution.

Taking the above issues in consideration, the aim of this thesis is to explore ways to perform electronic structure calculations with greater mathematical rigour, through the application of rigorous numerical methods. Of which, we focus in particular on methods based on interval analysis and deterministic global optimization. The Hartree-Fock electronic structure method will be used as the subject of this study due to its ubiquity within this domain.

We outline an approach for placing rigorous bounds on numerical error in Hartree-Fock computations. This is achieved through the application of interval analysis techniques, which are able to rigorously bound and propagate quantities affected by numerical errors. Using this approach, we implement a program called *Interval Hartree-Fock*. Given a closed-shell system and the current electronic state, this program is able to compute rigorous error bounds on quantities including i) the total energy, ii) molecular orbital energies, iii) molecular orbital coefficients, and iv) derived electronic properties.

Interval Hartree-Fock is adapted as an error analysis tool for studying the impact of numerical error in Hartree-Fock computations. It is used to investigate the effect of input related factors such as system size and basis set types on the numerical accuracy of the Hartree-Fock total energy. Consideration is also given to the impact of various algorithm design decisions. Examples include the application of different integral screening thresholds, the variation between single and double precision arithmetic in two-electron integral evaluation, and the adjustment of interpolation table granularity. These factors are relevant to both the usage of conventional Hartree-Fock code, and the development of Hartree-Fock

code optimized for novel computing devices such as graphics processing units.

We then present an approach for solving the Hartree-Fock equations to within a guaranteed margin of error. This is achieved by treating the Hartree-Fock equations as a non-convex global optimization problem, which is then solved using deterministic global optimization. The main contribution of this work is the development of algorithms for handling quantum chemistry specific expressions such as the one and two-electron integrals within the deterministic global optimization framework. This approach was implemented as an extension to an existing open source solver.

Proof of concept calculations are performed for a variety of problems within Hartree-Fock theory, including those in i) point energy calculation, ii) geometry optimization, iii) basis set optimization, and iv) excited state calculation. Performance analyses of these calculations are also presented and discussed.

Contents

List of Figures

List of Tables

Part

I

Introduction and Background

Chapter 1

Introduction

Contents

Electronic structure theory concerns the mathematical description of chemical systems based on quantum mechanics. Fundamental to this description is *Schrödinger's equation*, which when solved allows all atomic and molecular properties to be evaluated from first principles. For all practical purposes, however, Schrödinger's equation can only be solved approximately. This has led to the development of a variety of different *electronic structure methods*.

For each electronic structure method there is a different trade-off between the *quality* of the computed results and the *time to solution*. Traditionally the former has been interpreted to mean the level of theoretical approximation made when deriving the solution, but with the increasing capability of computer systems it is timely to consider a slightly broader definition of quality. In this respect this thesis considers two issues:

1. Electronic structure calculations are affected at the very least by *numerical errors*. This includes *rounding errors* due to the use of finite precision arithmetic, and *truncation errors* due to the application of numerical approximation. Although these errors appear insignificant when viewed in isolation, they can accumulate significantly especially in large scale calculations or in the presence of *numerical instability* leading to a loss of quality.

2. The majority of problems posed in electronic structure theory are equivalent to problems in non-convex global optimization. However, due to practical considerations, they are usually solved using iterative methods based on local optimization. These methods are not guaranteed to converge rigorously to the global minimum and the desired solution, unless a good initial guess is provided beforehand. In this case it is not just a matter of a loss of quality in an otherwise correct solution, but an erroneous solution.

The first issue is becoming increasingly relevant given the evolution of new massively multi-core devices and the push towards exascale computing. In the former this may result from the presence of a large performance difference motivating the use of lower levels of numerical precision than normally or through the use of non traditional rounding modes, while in the latter the sheer number of operations performed can result in unacceptably large numerical errors.

The second issue is a well known shortcoming of most electronic structure methods. Despite the advances in numerical algorithms in the past decades, ensuring convergence to a global minimum is still largely a manual process requiring the user to detect and rectify problems. Typically the approaches used are *ad-hoc*, in that they require the analyst to make intuitive, and at times arbitrary, adjustments to the underlying problem, from which another attempt at finding a solution is made. This is highly dependent on the knowledge and experience of the analyst, and becomes more difficult as larger problems, with high degrees of freedom, are considered.

These issues illustrate that, even when all theoretical constraints are satisfied, the results obtained from an electronic structure calculation need not be accurate due to the errors associated with the underlying numerical computations. On the other hand, it is also extremely difficult to accurately predict the margins of error involved given the complicated nature of electronic structure codes, and the non-transparent way in which errors are introduced and propagated within them. With respect to the two issues above, there is no mechanism in the floating point number system to keep track of the perturbations introduced by numerical errors, nor is there a straight forward approach to verify that an iterative method has successfully converged to a global minimum.

With these observations in mind, the goal of this thesis is to explore ways to perform electronic structure calculations with greater mathematical rigour, through the application of *rigorous methods*. Of these methods, we focus in particular on *interval analysis* and *deterministic global optimization*.

Interval analysis is a method of computation in which each uncertain quantity is represented by an interval that spans the worst-case estimate of its range. The

results of such a computation can then be expressed in terms of an interval that rigorously bounds the range of perturbations when all sources of uncertainty are taken into account.

Deterministic global optimization is a class of optimization techniques that are able to find solutions which are guaranteed to be optimal to within a specified threshold. This is usually achieved through a *rigorous search* over the entire feasible domain.

These methods are applied within the context of *Hartree-Fock theory*, a ubiquitous model from which many electronic structure methods are based. Hartree-Fock theory centres on a system of partial-differential equations known as the *Hartree-Fock equations*, which when solved yields an approximate solution to Schrödinger's equation. Although only a specific instance, the quality issues associated with Hartree-Fock computation are characteristic of broader issues faced by other electronic structure methods.

1.1 Contributions

This thesis makes the following major contributions to the two areas mentioned above.

1.1.1 Bounding Numerical Errors in Hartree-Fock Computation Using Interval Analysis

A standard Hartree-Fock code uses finite precision arithmetic and requires the application of a series of numerical approximations. The objective of this work is to compute rigorous bounds on the rounding and truncation errors that result. This was achieved by using interval analysis to represent each error contaminated quantity, and perform calculations between them. The significant contributions made include:

- The development of approaches for bounding truncation errors arising from the *series evaluation*, and *polynomial interpolation* operations used to compute approximate solutions to the *reduced incomplete gamma function* or $F_m(T)$. Where $F_m(T)$ is a key quantity used to construct *one and two-electron integrals*.
- The application of accurate summation techniques including *compensated summation* and *cascaded accumulated summation* to the evaluation of quantities including $F_m(T)$, the *Fock matrix*, and the *Hartree-Fock total energy*.

This allows the computation of tighter error bounds on these quantities by reducing the accumulation of rounding errors.

- The application of interval matrix analysis techniques to evaluate bounds on the eigenvalues and eigenvectors of Fock matrices consisting of uncertain elements affected by numerical errors. This is equivalent to diagonalizing an interval Fock matrix.

Using these methods, we implement an *Interval Hartree-Fock* program. Given an input specifying the molecular system, and the current electronic state, this program is able to generate rigorous error bounds on quantities including i) *the total energy*, ii) *the atomic orbital energies*, iii) *the molecular orbitals*, and iv) *derived electronic properties*.

In addition to providing error bounds, the interval Hartree-Fock program can also be used as a worst-case error analysis tool. In this thesis, we use it to study the effect of various input and design related factors on the quality of Hartree-Fock computation. These investigations are motivated by practical issues arising from conventional Hartree-Fock computation, as well as when performing Hartree-Fock computations in less conventional settings, for example on specialized processor hardware such as graphics processors. The following outlines particular contributions made in this pursuit:

- An investigation of numerical errors in $F_m(T)$ values due to the choice of series evaluation and polynomial interpolation scheme.
- A study analysing the accumulation of numerical errors in the Hartree-Fock total energy as a result of input related factors such as increasing *basis set* and system size. A polynomial regression model is proposed to predict errors in very large problems sizes, based on existing data.
- An analysis of numerical errors due to design related factors including i) the variation of integral pre-screening thresholds, ii) the application of accurate summation techniques, iii) the choice of $F_m(T)$ evaluation scheme, iv) the variation between single and double precision arithmetic in integral evaluation, and v) the variation of interpolation table granularity.

Partial details of the implementation and the error analysis results have been published in the following papers [58, 59]:

- P.P. Janes and A.P. Rendell, "Including Rigorous Numerical Bounds in Quantum Chemistry Calculations: Gaussian Integral Evaluation", Proceedings of

the 2008 IEEE 11th International Conference on Computational Science and Engineering, pages 75-82 (DOI 10.1109/CSE.2008.14).

- P.P. Janes and A.P. Rendell, "Placing Rigorous Bounds on Numerical Errors in Hartree-Fock Energy Computations", Journal of Chemical Theory and Computation, 2011 (6), 1631-1639

1.1.2 Solving the Hartree-Fock Equations using Deterministic Global Optimization

The solution to the Hartree-Fock equations can be expressed in terms of a non-convex global optimization problem by invoking the variational principle. However, it is typically solved using iterative procedures that are not guaranteed to converge towards the optimal solution unless a good initial guess is provided. When this is not the case, the results obtained can be entirely misleading.

With the aim of overcoming this issue, a program for solving closed-shell Hartree-Fock problems using deterministic global optimization has been developed. Unlike existing approaches, this is able to provide strong guarantees on optimality to within an arbitrary specified threshold ϵ, assuming that exact arithmetic is used.

The main contribution of this work is the development of algorithms for handling quantum chemistry specific mathematical expressions such as the one and two-electron integrals within the context of deterministic global optimization. This included:

- The development of two approaches for generating *linear relaxations* for the one and two-electron integrals. The first is based on decomposing the integrals into a linear combination of $F_m(T)$ functions and then generating a linear relaxation for each $F_m(T)$, the other is based on generating a linear relaxation for the two center $(ss|ss)$-type two-electron integrals using numerically determined parameters.
- The implementation of code for the *bounds tightening*, and *branch selection* of expressions involving the $F_m(T)$ and $(ss|ss)$ terms.

These approaches are implemented as an extension to COUENNE (*Convex Over and Under ENvelopes for Nonlinear Estimation*), an open source, general purpose, deterministic solver for mixed-integer non-linear programming (MINLP). A further modified version of COUENNE was also implemented for finding multiple global

minima. For brevity, we shall refer to our extended implementation of COUENNE as Hartree-Fock enabled COUENNE.

The proposed approaches are generalizable to almost every problem under Hartree-Fock theory. They enable applications in, i) *point energy calculation*, ii) *geometry optimization*, iii) *basis set optimization*, and iv) *excited state calculation*, amongst others. They can also be used to perform rigorous *computer assisted proofs* for theorems.

Proof of concept calculations are presented for a number of application areas, focusing in particular on geometry optimization. Additional experiments are also performed in order to characterize the performance of the solver. Contributions made in this area are summarized as follows

- A performance comparison between COUENNE against comparable deterministic MINLP solvers in solving non-linear programming benchmark problems, and the point energy calculation problem.
- A performance characterization of the Hartree-Fock enabled COUENNE code for geometry optimization problems; considering the impact of factors such as the choice of bounds tightening, linear relaxation, and branch selection algorithm.
- Proof of concept applications in basis set optimization and excited state calculation.

Partial details of the solver and the results for geometry optimization have been published in the following journal [57]:

- P.P. Janes and A.P. Rendell, "Deterministic Global Optimization in Ab-Initio Quantum Chemistry", Journal of Global Optimization (Accepted, February 2012), DOI: 10.1007/s10898-012-9868-5

1.2 Organization of Thesis

Chapter 2 presents background material that is relevant to the chapters that follows. It presents an overview of electronic structure theory, focusing in particular on Hartree-Fock theory. It also provides an introduction to topics including error analysis, interval analysis, and optimization.

The rest of the chapters, excluding the conclusion, are divided into two parts, one addressing the first research objective (Chapters 3 and 4), the other addressing the second research objective (Chapters 5 and 6).

Chapter 3 introduces methods for placing rigorous bounds on numerical errors in Hartree-Fock computation. It discusses the use of interval analysis techniques in generating error bounds at various steps in the calculation, focusing in particular on the $F_m(T)$ evaluation, and the Fock matrix diagonalization steps.

Chapter 4 proposes the use of the interval error bounding approach as tool for error analysis. This is applied to investigate numerical errors in Hartree-Fock computation due input and design related factors, focusing in particular on issues related to increasing basis set and system size, and those related to computational codes design for specialized processing hardware.

Chapter 5 outlines methods for evaluating the Hartree-Fock equations using deterministic global optimization techniques. It first provides the background to deterministic global optimization, then discusses specific methods for handling quantum chemistry specific mathematical expressions.

Chapter 6 applies the deterministic global optimization approach to problems in Hartree-Fock theory, and presents a detailed performance analysis.

Conclusions are given in Chapter 7 which also discusses future work.

Chapter

2

Background

Contents

2.1 Introduction

This chapter provides background material relevant to the research presented in this thesis. This includes a brief introduction to electronic structure theory that focuses on the Hartree-Fock method. This is followed by a discussion about the various sources of errors in numerical computing, and about interval analysis. The chapter concludes with a brief overview of some basic concepts in optimization.

2.2 Electronic Structure Theory

Electronic structure theory is concerned with accurately predicting chemical phenomena at the molecular scale. Quantum theory postulates that the behaviour of elementary particles such as electrons have both wave-like and particle-like characteristics. Any system composed of these particles can be fully described by a *wavefunction*, Ψ. While Ψ has no specific form, it is best described as a mathematical function that encodes all the characteristics of the system under consideration, such as the electron distribution, the electronic charge, and the electric and nuclear coordinates, etc. In general, Ψ is obtained by solving the *Schrödinger equation*, which in its non-relativistic, time independent form is given by

$$\hat{H}\Psi = E\Psi. \tag{2.1}$$

The *Hamiltonian* operator, $\hat{H}$, when applied to Ψ produces the product of the total energy E and Ψ. The Hamiltonian contains operations which define the interactions between the elementary particles. Figure 2.1 illustrates a simple two nuclei coordinate system with nuclei denoted by the indices A and B, and electrons denoted by the indices i and j. The coordinates of the A and B nuclei and the i and j electrons are defined respectively as $\mathbf{R}_A$, $\mathbf{R}_B$, r_i, and r_j. The distance between the i electron and the A nucleus is denoted as $r_{iA} = |r_i - R_A|$; the distance between the i and j electron is denoted as $r_{i,j} = |r_i - r_j|$, and the distance between the A and B nucleus is denoted as R_{AB}. Using these notations, the Hamiltonian for a generalized system of N_{atom} nuclei and N_{elec} electrons can be written as

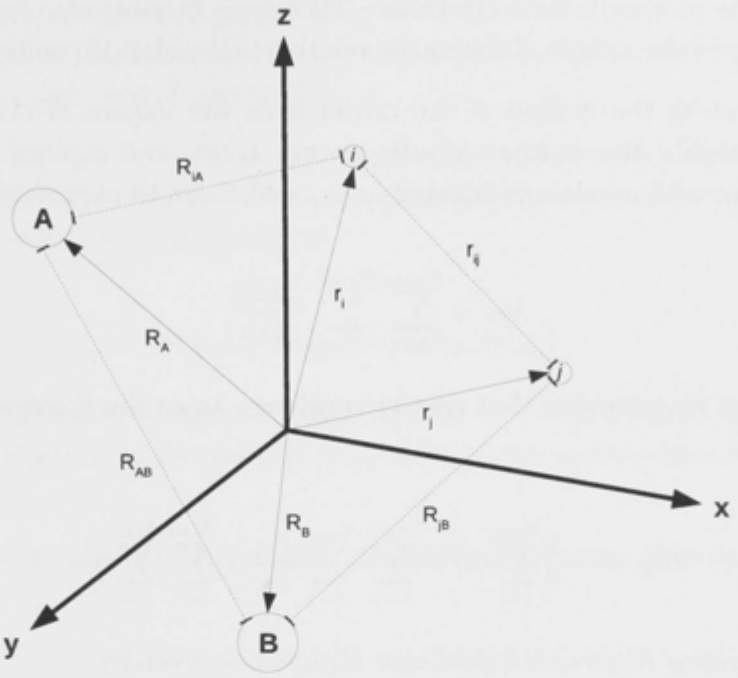


Figure 2.1: A molecular coordinate system consisting of two nuclei denoted as A and B, and two electrons denoted as i and j. Derived from [148].

$$\hat{H} = -\frac{1}{2}\sum_{i=1}^{N_{elec}} \nabla_i^2 - \sum_{A=1}^{N_{atom}} \frac{1}{2M_A}\nabla_A^2 - \sum_{i=1}^{N_{elec}}\sum_{A=1}^{N_{atom}} \frac{Z_A}{r_{iA}} + \sum_{A=1}^{N_{atom}}\sum_{B>A}^{N_{atom}} \frac{Z_A Z_B}{R_{AB}} + \sum_{i=1}^{N_{elec}}\sum_{j>i}^{N_{elec}} \frac{1}{r_{ij}} \tag{2.2}$$

where M_A is the ratio of the mass of nucleus A to the mass of an electron, and Z_A is the atomic number of nucleus A. ∇_i^2 and ∇_A^2 are Laplace operators with respect to the coordinates of the i electron and A nucleus, respectively. In summary, the Hamiltonian contains operations associated with the kinetic energy of electrons (first term); the kinetic energy of nuclei (second term); the Coulomb attraction between nuclei and electrons (third term); the repulsion between nuclei (fourth term), and the repulsion between electrons (fifth term).

2.2.1 Born-Oppenheimer Approximation

Schrödinger's equation is a Partial Differential Equation (PDE). Like most PDEs, an analytical solution to Schrödinger's equation is only possible for very simple systems. In order to solve larger systems it is necessary to apply a series of approximations. Primary amongst these is the *Born-Oppenheimer Approximation*; which derives from the observation that nuclei are over 1,800 times heavier and

move much more slowly than electrons. Therefore, in most circumstances it is reasonable to fix the motion of the nuclei relative to the electron motion.

By separating the motion of the nuclei from the motion of electrons, it is possible to negate the nuclear kinetic energy term, and replace the nuclear repulsion term with a constant denoted as V_{nn} which can be expressed as

$$V_{nn} = \sum_{A=1}^{N_{atom}} \sum_{B>A}^{N_{atom}} \frac{Z_A Z_B}{R_{AB}}. \tag{2.3}$$

The simplified Hamiltonian that results is referred to as the *Electronic Hamiltonian*

$$\hat{H}_{elec} = -\frac{1}{2} \sum_{i=1}^{N_{elec}} \nabla_i^2 - \sum_{i=1}^{N_{elec}} \sum_{A=1}^{N_{atom}} \frac{Z_A}{r_{iA}} + \sum_{i=1}^{N_{elec}} \sum_{j>i}^{N_{elec}} \frac{1}{r_{ij}}. \tag{2.4}$$

The corresponding *Electronic Schödinger Equation* is given by

$$\hat{H}_{elec} \Psi_{elec} = E_{elec} \Psi_{elec} \tag{2.5}$$

where the *electronic wavefunction*, Ψ_{elec}, describes the motion of electrons with respect to a field of fixed nuclei. E_{elec} denotes the electronic energy, which when added to V_{nn} produces an approximation of the total energy

$$E_{Total} = E_{elec} + V_{nn}; \tag{2.6}$$

Ψ_{elec} and E_{elec} depend explicitly on the electronic coordinates, and parametrically on the fixed nuclear coordinates. The solution of the electronic problem with respect to different nuclear coordinates defines a *potential energy surface*, in which the minima corresponds to geometries which are stable to some degree.

2.2.2 The Hartree-Fock Approximation

Although the Born Oppenheimer approximation greatly simplifies the Schrödinger Equation, the resulting problem is still too difficult to solve analytically except for simple systems. It is, however, in a form that is more tractable to numerical methods. The development of numerical methods for this purpose has been a major preoccupation of quantum chemists for the past fifty years. Central to the many attempts to solve the electronic Schrödinger's Equation is the *Hartree-Fock Approximation*. Although not the most accurate approach, it provides an important

theoretical basis from which many other, more advanced, approaches have been developed. Hartree-Fock theory, therefore, provides an appropriate starting point for research in electronic structure theory.

Hartree-Fock theory involves the approximate description of the electronic wave function Ψ_{elec} as an antisymmetric product of *one-electron trial functions* referred to as *molecular orbitals* (M.O),

$$|\Psi_{Elec}\rangle \approx |\Psi_{HF}\rangle = |\phi_1\phi_2...\phi_{N_{elec}}\rangle\,, \tag{2.7}$$

where each molecular orbital, denoted ϕ_μ, $\forall\mu \in \{1,2,..,N_{elec}\}$, describes the motion of an electron in the system. Ψ_{HF} denotes the approximate *Hartree-Fock wavefunction*.

Assuming that Ψ_{HF} is normalized, then the electronic *Hartree-Fock energy*, E_{elec}^{HF} is given as

$$E_{elec}^{HF} = \left\langle \Psi_{HF}|\hat{H}_{elec}|\Psi_{HF}\right\rangle, \tag{2.8}$$

where the term $\left\langle \Psi_{HF}|\hat{H}_{elec}|\Psi_{HF}\right\rangle$ denotes the inner product of Ψ_{HF} and $\hat{H}_{elec}\Psi_{HF}$.

The *variational principle* infers that the electronic energy of the approximate Hartree-Fock wave function is bounded from below by the exact electronic energy [148].

$$E_{elec} \leq E_{elec}^{HF}. \tag{2.9}$$

The above implies that the best approximate wavefunction is the one which yields the lowest energy with respect to a given functional space. This optimal wavefunction and its energy are referred to as the *ground state wavefunction* and the *ground state energy*, respectively.

The variational flexibility of the Hartree-Fock wavefunction lies in the definition of the molecular orbitals. In principle, molecular orbitals can be represented by any mathematical function, in practice, they are usually expressed in terms of a *linear combination of atomic orbitals* (LCAO),

$$\phi_\mu = \sum_{a=1}^{N} c_{\mu a}\chi_a \tag{2.10}$$

where χ_a are a set of N *atomic orbitals*, also referred to as *basis functions*, and $c_{\mu a}$ are the *molecular orbital coefficients*. The problem of finding the ground state

wavefunction, in Hartree-Fock theory, is therefore equivalent to that of finding the set of M.O coefficients that minimizes the electronic energy. The procedure to vary these coefficients is usually constrained by requiring orthonormality of the molecular orbitals,

$$\langle \phi_\mu | \phi_\upsilon \rangle = \delta_{\mu\upsilon}, \quad \forall \mu, \upsilon = \{1, 2, 3, .., N_{elec}\}. \tag{2.11}$$

2.2.3 The Hartree-Fock Equations

The *Hartree-Fock* equations is derived from the Lagrangian dual of the minimization of E_{elec}^{HF} (as defined in Equation 2.8) with respect to the set of M.O coefficients $\{c_{\mu a}\}$. When solved, this allows the determination of the ground state, or a higher energy state known as an *excited state*.

The Hartree-Fock equations for a system with N_{elec} electrons, and N basis functions can be expressed as a system of non-linear equations,

$$\sum_{b=1}^{N_{\text{elec}}} (F_{ab} - \varepsilon_i S_{ab}) c_{bi} = \mathbf{0} \quad \forall a = \{1, 2, .., N\} \tag{2.12}$$

or more commonly as a pseudo-eigenvalue problem, referred to as the *Roothaan Equation*,

$$\mathbf{FC} = \mathbf{SC}\varepsilon \tag{2.13}$$

where $\mathbf{F} \in \mathbb{R}^{N \times N}$ is the *Fock matrix*; $\mathbf{C} \in \mathbb{R}^{N \times N}$ is the matrix of molecular orbital coefficients, $\mathbf{S} \in \mathbb{R}^{N \times N}$ is the *overlap matrix* representing the area of overlap between atomic orbitals, and $\varepsilon \in \mathbb{R}^{N \times N}$ is a diagonal matrix of *molecular orbital energies*.

The elements of the overlap matrix are defined as

$$S_{ab} = (\chi_a | \chi_b). \tag{2.14}$$

The Fock matrix represents each electron moving in the average field of all other electrons. For a *closed shell* system (where N_{elec} is an even number), its elements are defined as

$$\begin{aligned} F_{ab} &= H_{ab}^{core} + \textstyle\sum_{c=1}^{N} \sum_{d=1}^{N} P_{cd} [(\chi_a \chi_b | \chi_c \chi_d) - \frac{1}{2} (\chi_a \chi_d | \chi_c \chi_b)] \\ &= H_{ab}^{core} + G_{ab} \end{aligned} \tag{2.15}$$

where the components of this expression originate from the elements of the electronic Hamiltonian operator in Equation 2.4 specifically. H_{ab}^{core} is an element of the *core-Hamiltonian* matrix, which incorporates the *one electron* kinetic energy and electron-nuclear attraction terms

$$H_{ab}^{core} = (\chi_a| - \frac{1}{2} \triangle |\chi_b) - \sum_{A=1}^{M} (\chi_a| \frac{Z_A}{r_{iA}} |\chi_b) \tag{2.16}$$

P_{cd} is an element of the *density matrix*, defined as

$$P_{cd} = 2 \sum_{i=1}^{N_{elec}/2} c_{ci} c_{di} \tag{2.17}$$

and $(\chi_a \chi_b | \chi_c \chi_d)$ are the *two-electron repulsion integrals* (ERI). ERIs are categorized into two types depending on the location of the indices a and b. $(\chi_a \chi_b | \chi_c \chi_d)$ referred to as the *Coulomb integral* represents the classical Coulomb repulsion between two electron distributions, while $(\chi_a \chi_d | \chi_c \chi_b)$ referred to as the *exchange integral* does not have a simple classical interpretation. The density matrix and the ERIs together form the *two-electron* term denoted as G_{ab}.

The electronic Hartree-Fock energy can be computed from the density matrix, the core-Hamiltonian matrix, and the Fock matrix using the following relation

$$E_{elec}^{HF} = \frac{1}{2} \sum_a \sum_b P_{ab} (H_{ab}^{core} + F_{ab}). \tag{2.18}$$

2.2.4 The Self-Consistent-Field Method

Since the elements of the Fock matrix depends non-linearly on the molecular orbital coefficients, the Roothaan equation (Equation 2.13) is usually solved by an iterative scheme known as the *Self-Consistent Field* (SCF) method.

The SCF method requires the transformation of the Roothaan equation into a matrix eigenvalue problem. This is usually achieved by computing a *transformation* matrix X [148], such that

$$\mathrm{X}^{\mathrm{T}} \mathrm{S} \mathrm{X} = \mathrm{I} \tag{2.19}$$

which leads to the following relation

$$\begin{aligned} \mathbf{FC} &= \mathbf{SC}\varepsilon \\ \mathbf{F}(\mathbf{X}\mathbf{X}^{-1})\mathbf{C} &= \mathbf{S}(\mathbf{X}\mathbf{X}^{-1})\mathbf{C}\varepsilon \\ (\mathbf{X}^{\mathrm{T}}\mathbf{F}\mathbf{X})(\mathbf{X}^{-1}\mathbf{C}) &= (\mathbf{X}^{\mathrm{T}}\mathbf{S}\mathbf{X})(\mathbf{X}^{-1}\mathbf{C})\varepsilon. \end{aligned} \tag{2.20}$$

Letting $\mathbf{F}' = \mathbf{X}^{\mathrm{T}}\mathbf{F}\mathbf{X}$ and $C' = \mathbf{X}^{-1}\mathbf{C}$ results in a matrix eigenvalue problem known as the *transformed Roothaan equations*

$$\mathbf{F}'\mathbf{C}' = \mathbf{C}'\varepsilon \tag{2.21}$$

where $\mathbf{F}'$ is referred to as the *transformed Fock matrix*, and $\mathbf{C}'$ is referred to as the *transformed coefficient matrix*.

The SCF method solves the Roothaan equations by repeatedly solving the transformed Roothaan equations, starting with an initial guess of $\mathbf{C_0}$, and stopping only when the solution converges towards a fixed point. For example, the k^{th} iteration involves: calculating F_k from C_{k-1}; calculating F'_k from F_k; diagonalizing F'_k to produce C'_k; and calculating C_k from C'_k.

Algorithm 1 The Self-Consistent Field (SCF) Method

1: Read molecular system information: a set of nuclear coordinates $\{R_A\}$, atomic numbers $\{Z_A\}$, number of electrons, and set of atomic orbitals $\{\chi_a\}$.
2: Calculate one and two-electron integrals then construct: S_{ab}, H_{ab}^{core}, and $(ab|cd)$.
3: Calculate X such that $\mathbf{X}^{\mathrm{T}}\mathbf{S}\mathbf{X} = \mathbf{I}$.
4: Obtain an initial guess of C, denoted as $\mathbf{C_0}$.
5: $k \leftarrow 0$
6: **repeat**
7: Calculate $\mathbf{P_k}$ from $\mathbf{C_k}$ using Equation 2.17.
8: Calculate $\mathbf{F_k}$ from $\mathbf{P_k}$ using Equation 2.15.
9: Calculate the transformed Fock matrix: $\mathbf{F'_k} = \mathbf{X}^{\mathrm{T}}\mathbf{F_k}\mathbf{X}$.
10: Diagonalize $\mathbf{F'_k}$ to obtain $\mathbf{C'_{k+1}}$ and ε_{k+1}.
11: Calculate $\mathbf{C_{k+1}} = \mathbf{X}\mathbf{C'_{k+1}}$
12: $k \leftarrow k + 1$
13: **until** Convergence.
14: If the procedure converges, then $\mathbf{C_{k+1}}$, $\mathbf{P_{k+1}}$, $\mathbf{F_{k+1}}$ etc. can be used to determine the ground state energy, population analyses, and other quantities of interest.

The simplest initial guess is a zero matrix $C_0 = 0$, which is equivalent to providing an initial guess that neglects the two-electron term from the Fock matrix.

The SCF procedure is not guaranteed to always converge towards the ground state, since the solution of the Roothaan equation could also correspond to an

undesired higher energy state. An SCF procedure may also fail because the M.O coefficient matrix oscillates infinitely between states, or diverge completely. This can be addressed to some degree by averaging the density matrix across successive iterations, or better still by using extrapolation schemes such as the *Direct Inversion of Iterative Subspaces* (DIIS) [119].

There are a number of possible convergence criterion's for the SCF procedure. One simple test which is often used requires that the electronic Hartree-Fock energy arising from C_k and C_{k+1} differ by no more than a small threshold, δ.

2.2.5 Basis Functions

In principle, the basis functions $\{\chi_a\}_{a=1}^{N}$ used to describe the Hartree-Fock molecular orbitals can be constructed from any arbitrary set of mathematical functions. Historically, the preferred representation was Slater-type orbitals (STOs) located on each atomic nucleus

$$\chi_a = r^{k-1} e^{-\zeta r} Y_l^m(\theta, \phi). \tag{2.22}$$

These functions have properties similar to those of exact atomic orbitals; in particular they exhibit a cusp at the nucleus and decay slowly as the distance from the nucleus increases. However, STOs are rarely used today as it is very computationally expensive to evaluate two-electron integrals that involve STOs [39, 13]. Instead most contemporary approaches prefer *Gaussian functions*, which are easier to integrate, and can be contracted together to mimic a Slater orbital. These *Contracted Gaussian functions* (CGF) denoted as χ_a are formed as linear combinations of *Primitive (Cartesian) Gaussian functions* (PGF) denoted as $\chi_{ak}(r)$,

$$\begin{aligned} \chi_a \equiv \quad \chi_a(r) \quad &= \textstyle\sum_{k=1}^{K_a} D_{ak} \chi_{a,k}, \quad \text{where} \\ \chi_{a,k}(r) \quad &= (X - X_a)^{x_a} (Y - Y_a)^{y_a} (Z - Z_a)^{z_a} e^{\alpha_{a,k} |r - R_a|^2} \\ &= g_{l_a}(\alpha_{a,k}, R_a) \end{aligned} \tag{2.23}$$

where K_a is the *degree of contraction*, $D_{a,i}$ a *contraction coefficient*, $R_a = (X_a, Y_a, Z_a)$ the coordinates where the orbital is centered, $l_a = \{x_a, y_a, z_a\}$ integers describing the Cartesian *angular components* of the orbital, and $\{\alpha_{a,i}\}$ the *orbital exponent*. Although, Gaussian functions do not exhibit the desired properties of Slater orbitals, a Slater orbital can be approximated to any desired accuracy by a CGF given sufficient numbers of appropriately defined PGFs [39].

Orbitals are categorized into types according to the total angular momentum $L_a = x_a + y_a + z_a$, where L_a values of 0, 1, 2, 3, 4 and 5 refer to s, p, d, f, and g

electron shells, respectively. An electron shell consists of $2L_a + 1$ *sub-shells* each corresponding to different permutations of x_a, y_a, z_a. A p_x shell, for instance, denotes the x-aligned sub-shell of the p shell, i.e. $l_a = \{1, 0, 0\}$. A sub-shell can be *occupied* by up to two electrons of opposing spin, therefore each shell has the maximum capacity of $2(2L_a + 1)$ electrons in total. An individual atomic orbital usually serves to represent an *occupied* or *unoccupied* sub-shell within the system being considered.

2.2.5.1 Gaussian Basis Sets

Today there are hundreds of well known CGF basis sets. Basis sets with few basis functions are generally more cost effective computationally, while basis sets with many basis functions generally yield more accurate results. The efficiency and accuracy of a basis set also depends on the degree of contraction of each basis function, and the parameters of each primitive Gaussian function.

The simplest Gaussian basis sets are derived from a least squares fit of each basis function to a Slater-type orbital. These basis sets are referred to as the STO-nG type *minimal basis sets*, where n denotes the degree of contraction of the basis functions. These basis sets are considered to be *minimal* because they contain only one CGF per occupied electron shell, hence providing the minimal representation of the system. While electronic structure calculations using minimal basis sets are relatively inexpensive, they are usually only accurate enough for a preliminary survey of the system being investigated.

Example 2.1. (Minimal Basis Sets) The ground state electronic configuration of Oxygen is $1s^2 2s^2 2p^4$, therefore the minimal basis set representation of Oxygen consists of two s-type CGFs, a p_x-type CGF, a p_y-type CGF, and a p_z-type CGF.

Basis sets larger than minimal basis sets are referred to as *extended basis sets*. They are mainly designed to account for bonding related factors such as orbital hybridization, polarization, and charge transfer. This adds greater variational flexibility with respect to the Hartree-Fock approximation. Some examples of commonly used extended basis sets include 3-21G, 4-31G, 6-31G, 6-31G**, 6-311++G(2d,1p), cc-pVDZ, cc-pVTZ, cc-pVQZ [137].

2.2.6 The Two-Electron Repulsion Integral

The two-electron repulsion integral (ERI) appearing in Equation 2.15 can be written as

$$(\chi_a\chi_b|\chi_c\chi_d) \equiv \iint \frac{\chi_a(r_1)\ \chi_b(r_1)\chi_c(r_2)\ \chi_d(r_2)dr_1dr_2}{|r_1 - r_2|} \tag{2.24}$$

$$(\chi_a\chi_b|\chi_c\chi_d) \equiv \sum_{i=1}^{K_a}\sum_{j=1}^{K_b}\sum_{k=1}^{K_c}\sum_{l=1}^{K_d} D_{a,i}D_{b,j}D_{c,k}D_{d,l}[\chi_{a,i}\chi_{b,j}|\chi_{c,k}\chi_{d,l}] \tag{2.25}$$

where the square brackets denote two-electron integrals over primitive Gaussian basis functions, or *primitive ERIs*, and the variables r_1 and r_2 denote the coordinates of the two electrons. It should be noted that there is an eight-way symmetry with respect to permutations of the basis functions which can be written as,

$$\begin{aligned}(\chi_a\chi_b|\chi_c\chi_d) = (\chi_b\chi_a|\chi_d\chi_c) = (\chi_c\chi_d|\chi_a\chi_b) = (\chi_d\chi_c|\chi_b\chi_a) = \\ (\chi_b\chi_a|\chi_c\chi_d) = (\chi_a\chi_b|\chi_d\chi_c) = (\chi_c\chi_d|\chi_b\chi_a) = (\chi_d\chi_c|\chi_a\chi_b).\end{aligned} \tag{2.26}$$

It can be seen from Equation 2.15 that up to $O(N^4)$ two-electron integrals are required in order to construct the Fock matrix (where N is the total number of atomic orbitals). For this reason, the task of computing, storing, and accessing ERIs is generally the most time consuming component of an SCF calculation.

In the following discussion, the notation for the ERIs are simplified, such that $(ab|cd)$ refers to $(\chi_a\chi_b|\chi_c\chi_d)$, and $[ab|cd]$ refers to a primitive integral in $(ab|cd)$. Furthermore, we denote integrals consisting of s-type PGFs as $[ss|ss]$, p-type PGFs as $[pp|pp]$, d-type PGFs as $[dd|dd]$, and so on. This definition only distinguishes between the different Cartesian components of the PGFs when necessary.

2.2.7 Two-Electron Repulsion Integrals of s-functions

The simplest of the $[ss|ss]$-type integrals can be expressed as

$$I^{ERI} = \iint e^{-\alpha|r_1-\mathbf{A}|^2}e^{-\beta|r_1-\mathbf{B}|^2|}\frac{1}{|r_1 - r_2|}e^{-\gamma|r_2-\mathbf{C}|^2}e^{-\delta|r_2-\mathbf{D}|^2}dr_1dr_2 \tag{2.27}$$

where A, B, C, and D are the centers, and α, β, γ, δ the exponents of the four Gaussian functions. By the application of the *Gaussian product rule*, the four function integral can be rewritten as a two function integral,

$$I^{ERI} = G_{AB}G_{CD}\iint e^{-\zeta|r_1-\mathbf{P}|^2}\frac{1}{|r_1 - r_2|}e^{-\eta|r_2-\mathbf{Q}|^2}dr_1dr_2 \tag{2.28}$$

where

$$\begin{aligned} G_{AB} &= e^{\frac{-\alpha\beta}{\zeta}|\mathbf{A}-\mathbf{B}|^2} & G_{CD} &= e^{\frac{-\gamma\delta}{\eta}|\mathbf{C}-\mathbf{D}|^2} \\ \mathbf{P} &= \frac{\alpha A + \beta B}{\alpha + \beta} & \mathbf{Q} &= \frac{\gamma C + \delta D}{\gamma + \delta} \\ \zeta &= \alpha + \beta & \eta &= \gamma + \delta. \end{aligned} \tag{2.29}$$

By replacing the remaining terms in the integrand by their Fourier representations (and applying a number of other mathematical transformations [39]), the two function integral can be reduced further to a one function integral,

$$I^{ERI} = G_{AB} G_{CD} \frac{2\pi^{5/2}}{\zeta\eta(\zeta+\eta)^{1/2}} F_0(T) \tag{2.30}$$

where

$$\begin{aligned} \upsilon^2 &= \frac{\zeta\eta}{\zeta+\eta} \\ \mathbf{R} &= \mathbf{Q} - \mathbf{P} \\ T &= \upsilon^2 R^2 \\ F_0(T) &= \int_0^1 e^{-Tt^2} dt. \end{aligned} \tag{2.31}$$

The evaluation of a four centre two-electron repulsion integral involving s functions therefore reduces to a problem of evaluating $F_0(T)$, which can be expressed more generally as a *reduced incomplete gamma function* or $F_m(T)$, of the form

$$F_m(T) = \int_0^1 t^{2m} e^{-Tt^2} dt \tag{2.32}$$

where $m \in \mathbb{Z}^+$ values larger than 0 are required in order to evaluate integrals with higher total angular momentum. The value of $F_m(T)$ itself can only be determined analytically when $T = 0$, otherwise it must be approximated numerically.

2.2.8 Notations

Before more complicated integrals are discussed, it is necessary to introduce some additional notations. As seen earlier with $[ss|ss]$ integrals, it is not necessary for a two-electron integral to have four functions. Hence, $[abp|cdq]$ refers to a six function integral, $[p|q]$ a two function integral, and $[r]$ a one function integral. The function defined by the first or the second electron of a two-electron integral are referred to as the *bra*, or *ket* integrals, respectively. The bra integral of $[ab|cd]$ is denoted

as $[ab|$, and the ket integral as $|cd]$. The bra and ket notation is also generalizable to contractions, and two-electron integrals with different number of functions. For instance,

- $(r]$ - is a bra contracted integral
- $[r)$ - is a ket contracted integral
- $[p|$ - is the bra integral of $[p|q]$
- $|q]$ - is the ket integral of $[p|q]$

Contracted integrals are formed from a summation of primitive integrals during the *contraction step*. The bra and ket sides of the primitive integrals can be contracted separately as follows

Bra contraction $[ab|cd] \rightarrow (ab|cd]$:

$$(ab|cd] = \sum_{i=1}^{K_a} \sum_{j=1}^{K_b} [\chi_{a,i}\chi_{b,j}|\chi_c\chi_d). \tag{2.33}$$

Ket contraction $(ab|cd] \rightarrow (ab|cd)$:

$$(ab|cd) = \sum_{k=1}^{K_c} \sum_{l=1}^{K_d} (\chi_a\chi_b|\chi_{c,k}\chi_{d,l}]. \tag{2.34}$$

Bra and Ket contraction can be performed in any order.

2.2.9 Two-Electron Repulsion Integrals of Higher Angular Momentum

The previous section considered the evaluation of a simple $[ss|ss]$-type integral. This section considers the evaluation of integrals with higher total angular momentum such as $[pp|pp]$, $[dd|dd]$, etc. In general, these ERIs can be recursively decomposed into a linear combination of $[ss|ss]^m$ type integrals, where the superscript relates to the value of the index m in the expression $F_m(T)$. Example 2.35 illustrates such a decomposition for $[pp|pp] \equiv [pp|pp]^0$.

Example 2.2. (Decomposition of $[pp|pp]$)

$$\begin{aligned}
[pp|pp]^0 &= C_0[pp|ps]^0 + C_1[pp|ps]^1 + C_2[sp|ps]^1 + C_3[ps|ps]^1 + \\
&\quad C_4[pp|ss]^0 + C_5[pp|ss]^1 \\
[pp|ps]^1 &= C_0[pp|ss]^1 + C_1[pp|ss]^2 + C_2[sp|ss]^2 + C_3[ps|ps]^2 \\
[pp|ss]^2 &= C_0[ps|ss]^2 + C_1[ps|ss]^3 + C_2[ss|ss]^3 + C_3[ss|ss]^2 \\
[ps|ss]^3 &= C_0[ss|ss]^3 + C_1[ss|ss]^4
\end{aligned} \tag{2.35}$$

where C_0, C_1, C_2, C_3, C_4, and C_5 are scalar constants in $\mathbb{R}$. This decomposition is in fact based on the Obara-Saika (OS) [115] recursive relation, which is discussed later in this section.

In the general instance, the schema by which an integral is decomposed into integrals of lower angular momentum relies on the use of various *recursive relations* (RR). The most popular recursive relation schemes used today are those due to McMurchie and Davidson (MD) [103], Obara and Saika (OS) [115], Rys, Dupuis, and King (RDK) [128], and Head-Gordon and Pople (HGP) [44].

It should be noted that in a standard calculation, the recursive relations are not applied explicitly to $[ab|cd]$ for computational reasons. Instead $[ab|cd]$ is constructed sequentially, starting first from the most basic lower angular momentum integral.

2.2.9.1 The McMurchie-Davidson Algorithm

The McMurchie-Davidson (MD) algorithm [103] was one of the earliest recursive methods used for evaluating ERI, and was influential in the subsequent development of the OS [115] and HGP [44] algorithms.

At its core, the MD scheme requires the evaluation of $[0]^m$ integrals, related to the one function $[ss|ss]$-type integral given in Equation 2.27. $[0]^m$ is computed from $F_m(T)$ in the following *transformation* step, $F_m(T) \rightarrow [0]^m$:

$$[0]^m = D_a D_b D_c D_d G_{AB} G_{CD} \frac{2^{1/2}\pi^{\frac{5}{2}}}{(\zeta\eta)^{\frac{3}{2}}} (2\vartheta^2)^{m+1/2} F_m(T) \tag{2.36}$$

where the values of G_{AB}, G_{CD}, ζ, η, and ϑ are defined in Equation 2.29.

The transformation of the $[0]^m$ integrals to the $[r]$ integrals is achieved as follows, $[0]^m \rightarrow [r]$:

$$[r]^m = R_i[r - 1_i]^{m+1} - (r_i - 1)[r - 2_i]^{m+1} \tag{2.37}$$

where the label $i \in \{x, y, z\}$ denotes a particular Cartesian component, r is centered at **R** (Equation 2.31), R_i is the i^{th} component of **R** , and r_i is the i^{th} component of the angular momentum of r. $[r - 1_i]$ is equivalent to $[r]$ with its i^{th} angular momentum component reduced by 1. Note also that $[0]^m \equiv [r]^0$. These notations are applied to all recursion schemes discussed from this point forward.

The transformation from the $[r]$ integral to the $[p|q]$ integral is given by, $[r] \rightarrow [p|q]$:

$$[p|q] = (-1)^q[p + q] = (-1)^q[r] \tag{2.38}$$

where p and q are products of a, b and c, d, respectively. p and q are also Hermite functions centered on **P** and **Q** (Equation 2.29), respectively.

The $[ab|q]$ integrals are formed from $[p|q]$ using a bra transformation which shifts angular momentum from p to a and b, $[p|q] \rightarrow [ab|q]$:

$$[(a + 1_i)bp| = p_i[ab(p - 1_i)| + (P_i - A_i)[abp| + (2\zeta)^{-1}[ab(p + 1_i)| \tag{2.39}$$

where $[ab0|q]$ is equivalent to $[ab|q]$ where p has zero angular momentum, and ζ is defined in Equation 2.29.

Finally, the $[ab|cd]$ integrals are formed from the $[ab|q]$ integrals by an equivalent ket transformation which shifts angular momentum from q to c and d, $[ab|q] \rightarrow [ab|cd]$:

$$|(c + 1_i)dq] = q_i|cd(q - 1_i)] + (Q_i - C_i)|cdq] + (2\eta)^{-1}|cd(q + 1_i)]. \tag{2.40}$$

The bra and ket transformation steps can be applied in any order. Furthermore, the bra and ket contractions can be applied at any point after the bra and ket transformation, respectively. The original McMurchie-Davidson algorithm proposes the following order to construct $(ab|cd)$:

Equation 2.37	$[0]^m$	$\rightarrow$	$[r]^m$
Equation 2.38	$[r]^m$	$\rightarrow$	$[p|q]$
Equation 2.39	$[p|q]$	$\rightarrow$	$[ab|q]$
Bra Contraction	$[ab|q]$	$\rightarrow$	$(ab|q]$
Equation 2.40	$(ab|q]$	$\rightarrow$	$(ab|cd]$
Ket Contraction	$(ab|cd]$	$\rightarrow$	$(ab|cd)$

2.2.9.2 The Obara-Saika and Head-Gordon-Pople Algorithms

In 1985, Obara and Saika (OS) [115] specified an 8-term recursive relation for directly transforming $[ss|ss]^m$ type integrals in to $[ab|cd]$. This is considerably simpler than the formulation given by the McMurchie Davidson Algorithm which required a number of transformations to reach the same point. In OS the $[ss|ss]^m$ type integrals are denoted as $[00|00]^m$

$$[00|00]^m = D_a D_b D_c D_d G_{AB} G_{CD} \frac{2\pi^{\frac{5}{2}}}{\zeta\eta(\zeta+\eta)^{1/2}} F_m(T). \tag{2.41}$$

Obara-Saika did not clearly discuss how their recursive relation was to be applied. This task was instead completed by Head-Gordon and Pople (HGP) [44] who expressed the OS 8-term RR as two secondary recursive RRs called the *Vertical Recursive Relation* (VRR) and the *Horizontal Recursive Relation* (HRR).

The VRR transforms $[00|00]^m$ to $[m0|n0]$ as follows, $[00|00]^m \rightarrow [m0|n0]$

$$\begin{aligned}[(m+1_i)0|n0]^m \;=\; & (B_i - A_i)(\tfrac{2\beta}{2\zeta})[m0|n0]^m + R_i(\tfrac{1}{2\zeta})[m0|n0]^{m+1} \\ & + m_i\Big\{(\tfrac{1}{2\zeta})[(m-1_i)0|n0]^m - (\tfrac{1}{2\zeta})^2[(m-1_i)0|n0]^{m+1}\Big\} \\ & + n_i(\tfrac{1}{2\zeta})(\tfrac{1}{2\eta})[m0|(n-1_i)0]^{m+1}.\end{aligned} \tag{2.42}$$

The transformation from the $[m0|n0]$ integrals to the $[ab|n0]$ integrals is achieved by applying the HRR, $[m0|n0] \rightarrow [ab|n0]$:

$$[a(b+1_i)|cd] = [(a+1_i)b|cd] + (A_i - B_i)[ab|cd]. \tag{2.43}$$

The bra and ket transformations can be computed in any order. HRR can also be applied to contracted integrals. Therefore $(ab|cd)$ can be formed by first contracting $[m0|n0]$ and then applying HRR.

The original HGP can be used to construct $(ab|cd)$ in the following sequence:

VRR:	$[00\|00]^m$	$\rightarrow$	$[m0\|n0]$
Contract Bra:	$[m0\|n0]$	$\rightarrow$	$(m0\|n0]$
Contract Ket:	$(m0\|n0]$	$\rightarrow$	$(m0\|n0)$
Transfer Bra:	$(m0\|n0)$	$\rightarrow$	$(ab\|n0)$
Transfer Ket:	$(ab\|n0)$	$\rightarrow$	$(ab\|cd)$

While a modified version of OS suggested by Lindh [124], adopts the following sequence:

VRR:	$[00\|00]^m$	$\rightarrow$	$[m0\|n0]$
Transfer Bra:	$[m0\|n0]$	$\rightarrow$	$[ab\|n0]$
Transfer Ket:	$[ab\|n0]$	$\rightarrow$	$[ab\|cd]$
Contract Bra:	$[ab\|cd]$	$\rightarrow$	$(ab\|cd]$
Contract Ket:	$(ab\|cd]$	$\rightarrow$	$(ab\|cd)$

Note that the implementation of HGP is identical to that of OS, except for the fact that the contraction steps are applied earlier in HGP than in OS.

2.2.10 The PRISM Algorithm

The notion that it is possible to vary the sequence in which the various transfer and contraction operations are performed provides the basis for the PRISM algorithm [40]. PRISM is, in essence, a series of modifications to the MD and HGP algorithms that allows dynamic flexibility as to when the transformation and contraction steps are applied. This is primarily motivated by the observation that the cost of computing highly contracted integrals can be reduced considerably if the contraction steps are applied *early*, whereas mildly contracted integrals are more efficiently computed when the contraction is applied *late*.

In general $(ab|cd)$ is computed from three *transformation* (T) steps determined by a recursive relation and two *contraction* (C) steps. A particular order or *path* in which the integral is formed can be designated for example as, TTCTC, TTTCC, CCTTT, etc. With the optimal path determined dynamically based on the characteristics of the ERI being computed [40].

The PRISM algorithm for MD is illustrated in Figure 2.2.

2.3 Errors in Numerical Computing

All computational models are subject to various degrees of uncertainty. These uncertainties can be attributed to

- *Modelling errors*, introduced during the model building process; mainly due to the various abstractions and approximations made towards the original problem in order to obtain a mathematical model that is computable. In electronic structure methods, examples of modelling errors include the errors due to applying the Born-Oppenheimer, and the Hartree-Fock approximations. The convergence of SCF towards an undesired higher energy state due to the

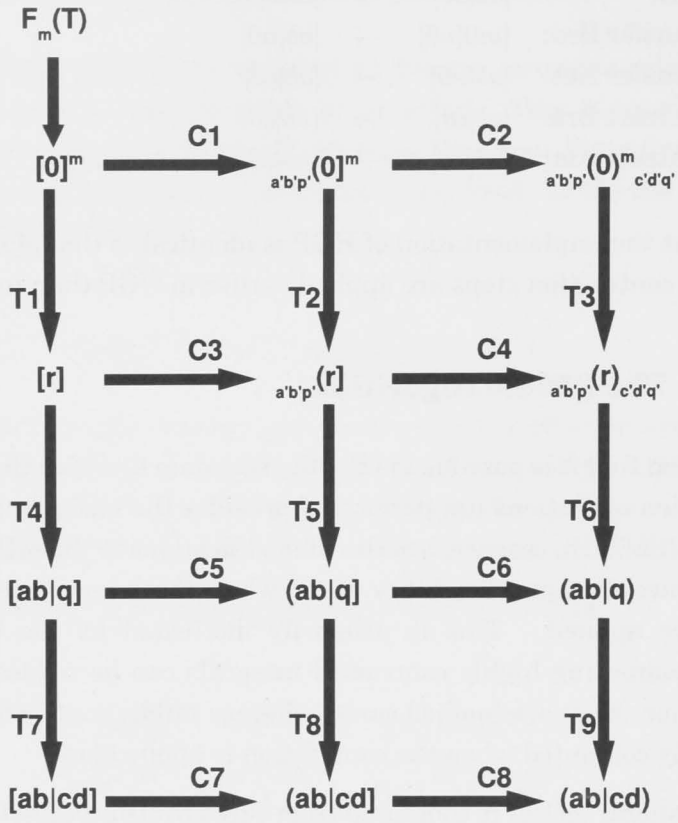


Figure 2.2: The McMurchie-Davidson PRISM.

formulation of the Roothaan equations can also be considered as a modelling error.

- *Data errors*, due to imprecise data provided as input to the model. This often occurs when the data is derived from empirical measurements, or results obtained from previous computations - both of which may have limited accuracy.

- *Numerical errors*, introduced when computing a numerical solution to the model; this includes *rounding errors*, *truncation errors*, *ill-conditioning*, and *instability* .

Errors can be expressed in terms of both their *absolute* and *relative* values.

Definition 2.3. (Absolute Error and Relative Error) Suppose that x and $\tilde{x}$ are the *exact* and *approximate* values of a quantity, respectively. Then the *absolute error* is given by

$$Absolute\ Error = |x - \tilde{x}|$$

whereas the *relative error* is given by

$$Relative\ Error = \left| \frac{x - \tilde{x}}{x} \right|$$

The following sections focuses on the different sources of numerical error, including truncation errors, rounding errors, ill-conditioning, and instability.

2.3.1 Truncation Errors

Truncation error or *approximation error* is the difference between the true result and the result that is produced by a given algorithm in exact arithmetic. There are numerous causes of truncation error. For example, it can be due to

- Evaluating an infinite series up to a finite point, e.g.

$$\sum_{i=1}^{\infty} f_i(x) \approx \sum_{i=1}^{N} f_i(x)$$

- Using a finite difference approximation to compute derivatives.
- Approximating the area of a function using N uniformly spaced trapezoids.
- Terminating an iterative method within a convergence threshold, ϵ. e.g.

$$f(x) \approx \tilde{f}(x), \text{ if } \|f(x) - \tilde{f}(x)\| \leq \epsilon$$

Two concrete examples are given below to further illustrate truncation errors.

Example 2.4. Consider the Taylor expansion of e^x centered at $a = 0$

$$e^x \approx 1 + x + \frac{1}{2!}x^2 + \frac{1}{3!}x^3 + ...$$

the error of this approximation is given by the *Lagrange Remainder Theorem* which states that there exists a number $\eta \in \mathbb{R}$ between x and a such that

$$\begin{aligned} Err &= \frac{f^{n+1}(\eta)}{(n+1)!}(x-a)^{n+1} \\ &= \frac{e^{\eta}}{(n+1)!}x^{n+1} \end{aligned}$$

where n is the number of terms included in the expansion. Thus, the truncation error decreases as the number of terms increases, and as $x \to a$. For example, the relative error of the expansion for e^x, $x = 1$ as a function of the number of terms n is given as follows

n	Rel Error
1	0.2642411
2	0.0803014
3	0.0189882
4	0.0036598
5	0.0005942
6	0.0000832
7	0.0000102
8	0.0000011

Example 2.5. Consider the first-order finite difference approximation of $sin'(x)$

$$sin'(x) \approx \frac{sin(x+h) - sin(x)}{h}$$

the error of this approximation decreases as the size of the discretization h decreases. The following table illustrates this relationship for $sin'(x)$, $x = 1$ as follows in terms of relative error

h	Rel Error
1×10^{0}	0.8744658
1×10^{-1}	0.0794713
1×10^{-2}	0.0078036
1×10^{-3}	0.0007789
1×10^{-4}	0.0000779
1×10^{-5}	0.0000078
1×10^{-6}	0.0000008

Another way to reduce the truncation error is by using a higher order approximation.

2.3.2 Rounding Errors

The vast majority of scientific calculations are performed using finite precision floating point arithmetic. As such, only a finite number of quantities can be

represented exactly. Rounding is used when an exact representation is not possible. There are several different *rounding modes* including

- round to nearest
- round upwards (towards ∞),
- round downwards (towards -∞), or
- round toward zero

The difference between the exact real number $x \in \mathbb{R}$ and its floating point representation, denoted as $fl(x)$, is known as the *rounding error*.

The *machine epsilon*, denoted as ϵ_{mach}, represents the maximum relative error due to rounding error for a given floating point number system. Assuming that the round to nearest scheme is used, ϵ_{mach} can be written as

$$\epsilon_{mach} = \beta^{1-t}/2 \tag{2.44}$$

where β is the base of the number system used, and t is the size of the *mantissa*. An IEEE754 standard double precision binary floating point number has a 52-bit mantissa, therefore $\epsilon_{mach} = 2^{-52} = 2.220446 \times 10^{-16}$.

We can express $fl(x)$ in terms of x as

$$fl(x) = x + \delta_x x, \quad \text{where } |\delta_x| \leq \epsilon_{mach}. \tag{2.45}$$

Rounding errors are propagated by arithmetic operations $\{+, -, \times, /\}$ between floating point numbers. Consider the floating point summation between two positive real numbers $a \in \mathbb{R}^+$ and $b \in \mathbb{R}^+$

$$\begin{aligned} fl(a) + fl(b) &= a + \delta_a a + b + \delta_b b \\ &\approx (a + b) + \delta(a + b) \quad \text{where } \delta = \max(\delta_a, \delta_b) \end{aligned}$$

the relative rounding error arising from *floating point addition* is in the order of ϵ_{mach}, since $|\delta| \leq \epsilon_{mach}$. Rounding errors become an issue mainly when it is accumulated over many operations. This is illustrated as follows in the case of *floating point summation*.

Example 2.6. Consider the summation

$$S_N = \sum_{i=1}^{N} x_i$$

where $x_i \in \mathbb{R}$. Letting $x_i = 0.1$, $\forall i \in \{1, 2, .., N\}$, the relative rounding errors as N increases is given in the following table

N	Rel Error
10	1.11×10^{-16}
100	1.95×10^{-15}
1,000	1.41×10^{-14}
10,000	1.59×10^{-13}
100,000	1.88×10^{-12}
1,000,000	1.33×10^{-11}
10,000,000	1.61×10^{-10}
100,000,000	1.89×10^{-9}
1,000,000,000	1.25×10^{-8}

With around 16 decimal digits of precision, it would appear that double precision floating point provides a substantial margin for obtaining accurate results despite rounding errors. However, on parallel computers capable of dispatching up to the order of 10^{15} floating point operations per second, the gradual accumulation of rounding errors alone may become significant enough to challenge this assumption.

Convention suggests that issues with rounding errors can be effectively addressed by further increasing the floating point precision. Furthermore it is thought that if the results of a calculation agree significantly with another set of results calculated with higher precision floating point arithmetic then the results are correct. An often cited example by Rump [127] (as modified by [91]) shows, however, that this may not always hold true.

Example 2.7. (Rump's Example) Letting $x_0 = 77617$, and $y_0 = 33096$, and

$$f(x, y) = (333.75 - x^2)y^6 + x^2(11x^2y^2 - 121y^4 - 2) + 5.5y^8 + \frac{x}{2y}. \tag{2.46}$$

Using IEEE754 floating point arithmetic and round to nearest, we obtain the following results for the above expression,

$$
\begin{aligned}
32-bit \quad & f(x_0, y_0) = 1.172604 \\
64-bit \quad & f(x_0, y_0) = 1.1726039400531786 \\
128-bit \quad & f(x_0, y_0) = 1.172603940053178631858834904520183 8
\end{aligned}
$$

The significant digits of the three results agree with each other, and suggest smooth convergence as precision increases. These results are, however, completely incorrect, even the sign is wrong. The true result is in fact

$$f(x_0, y_0) = -0.827396059946821368141165095479816..$$

Rump's example illustrates how a particularly severe case of numerical instability [91], known as *catastrophic cancellation*, can cause a total loss of precision even in 128-bit arithmetic. It also shows that, by using floating point arithmetic, regardless of how many bits of precision, one can be completely oblivious to this issue.

2.3.3 Conditioning and Stability

The *conditioning* of a mathematical problem refers to the sensitivity of the problem to errors in the input data [46]. A problem is said to be *well-conditioned* if small perturbations in the input result only in small perturbations in the solution. On the other hand, a problem is said to be *ill-conditioned* if small perturbations in the input causes large perturbations in the solution. An ill-conditioned problem is almost unsolvable in practice as it refers to a limitation inherent to the underlying mathematical model.

Central to determining the conditioning of a problem is the evaluation of the *condition number*, $\hat{\kappa}$. For the purpose of the following discussion, it is assumed that the problem refers to a mathematical function f.

Definition 2.8. (Condition Number) For a differentiable function $f : \mathbb{R}^n \to \mathbb{R}$ at $x \in \mathbb{R}^n$, the condition number of f is defined as

$$\hat{\kappa} = \sup_{\delta x} \frac{\|\partial f\|}{\|\partial x\|} = \|J(x)\| \tag{2.47}$$

where $J_{ij}(x) = \partial f_i / \partial x_j$ are elements of the Jacobian matrix, and the matrix norm is induced by the vector norms of ∂f and ∂x. The value of the condition number $\hat{\kappa}$ depends on the choice of norm. A small condition number indicates that the problem is well conditioned, whereas a large condition number indicates that the problem is ill-conditioned.

The *stability* of a computational algorithm refers to the sensitivity of the algorithm to numerical errors [45]. An algorithm is said to be *unstable* if perturbations caused by numerical errors are further propagated and amplified on to the solution. There are two relevant definitions of stability: *stability* and *backward stability*

Definition 2.9. (Stability) An algorithm $\tilde{f}$ for computing a function f is stable for all x if

$$\frac{\|\tilde{f}(x) - f(\tilde{x})\|}{\|f(\tilde{x})\|} = O(\epsilon_{mach}) \tag{2.48}$$

for some perturbed input $\tilde{x}$ with

$$\frac{\|\tilde{x} - x\|}{\|x\|} = O(\epsilon_{mach}). \tag{2.49}$$

In other words, *a stable algorithm provides nearly the right answer* (as in Equation 2.49) to nearly the right question (as in Equation 2.48).

Definition 2.10. (Backward Stability) An algorithm $\tilde{f}$ for computing a function f is backward stable if for all x,

$$\tilde{f}(x) = f(\tilde{x}) \tag{2.50}$$

for some perturbed input $\tilde{x}$ with

$$\frac{\|\tilde{x} - x\|}{\|x\|} = O(\epsilon_{mach}) \tag{2.51}$$

in other words, *a backward stable algorithm provides the right answer to nearly the right question.*

Cancellation error, as seen in Rump's example (Equation 2.46), is one of the most common causes of instability in numerical code. It occurs in situations where two numbers of the same sign and of similar magnitude are subtracted. If one or both of these numbers are subjected to numerical errors, then the subtraction has the effect of stripping off most of the significant digits, leaving behind a result that mainly consists of error contaminated digits.

The concept of stability of a computational algorithm is analogous to conditioning of a mathematical problem in that both relate to the effects of perturbations [45]. To solve a problem accurately, the problem must be well-conditioned, and the

algorithm used to solve it must be stable [45]. In contrast, significant inaccuracy can result when a stable algorithm is used to solve an ill-conditioned problem, or when an unstable algorithm is used to solve a well-conditioned problem.

2.4 Interval Analysis

In 1959 [108], Ramon Moore outlined a new arithmetic system based on the representation of numerical quantities in terms of *Real Intervals*: a bounded, closed subset of real numbers defined as

$$X = [\underline{X}, \overline{X}] = \{x \in \mathbb{R} \mid \underline{X} \leq x \leq \overline{X}\}$$

where $\underline{x}, \overline{x} \in \mathbb{R}$. We use $I(\mathbb{R})$ to denote the set of such intervals.

Definition 2.11. The following attributes of intervals are important for subsequent discussions

$$\begin{array}{lll} \textbf{(Midpoint)} & m(X) & = \frac{1}{2}(\underline{X} + \overline{X}) \\ \textbf{(Radius)} & r(X) & = \frac{1}{2}(\overline{X} - \underline{X}) \\ \textbf{(Magnitude)} & |X| & = \max_{x \in X} |x| = \max\{|\underline{X}|, |\overline{X}|\} \\ \textbf{(Width)} & w(X) & = \overline{X} - \underline{X}. \end{array} \tag{2.52}$$

An interval X where $\underline{x} = \overline{x}$, and therefore $w(X) = 0$, is known as a *degenerate interval*.

Intervals were originally proposed as a method for bounding rounding errors [108], the effect of which was of great concern due to the limited bits of precision available on computers at that time. Intervals can in fact be used to represent any *uncertain quantity* - this can potentially include values that are uncertain due to any combination of modelling errors, rounding errors, truncation errors, measurement errors, ill-conditioning, instability, unknown or poorly understood phenomena, and so on.

For example, the floating point representation of $x \in \mathbb{R}$, denoted as $fl(x)$, can be expressed in intervals as $[fl(x) - \delta, fl(x) + \delta]$ where $\delta = \epsilon_{mach} x$ to account for rounding error. Furthermore, suppose that a thermometer indicates that the temperature is 25°C and it is known that this particular thermometer is accurate to within 2°C, then the temperature can be represented in intervals as $[23, 27]$°C to account for the margin of error.

Definition 2.12. (Interval Vector and Matrices) A *Real Interval Vector* $V \in I(\mathbb{R}^n)$ is an n-dimensional vector consisting of interval elements $V_i \in I(\mathbb{R})$, $\forall i \in \{1, 2, .., n\}$. An interval vector can be used to denote a domain spanned by a closed hyperrectangle in Euclidean space.

Furthermore, a *Real Interval Matrix* $M \in I(\mathbb{R}^{m \times n})$ is an $m \times n$ matrix consisting of interval elements $M_{ij} \in I(\mathbb{R})$, $\forall i \in \{1, 2, .., m\}$ and $\forall j \in \{1, 2, .., n\}$.

The following sections will now introduce the concepts of interval arithmetic and interval functions.

2.4.1 Interval Arithmetic

Definition 2.13. (Containment Property) the set of basic arithmetic operations $\bullet \in \{+, -, \times, /\}$ between real intervals $X, Y \in I(\mathbb{R})$ are defined such that

$$X \bullet Y = \{x \bullet y \mid x \in X, y \in Y\}. \tag{2.53}$$

That is, the arithmetic operations between two intervals X and Y results in another interval that *contains* the results of operations between all possible combination of real values in X and Y. This property is generally referred to as the *containment property* for arithmetic operations [43].

We define *Real interval arithmetic* to be arithmetic operations between two intervals that, i) satisfies the containment property, and ii) carried out using exact arithmetic.

Definition 2.14. (Real Interval Arithmetic) Addition, subtraction, multiplication, and division operations between real intervals are defined as follows

$$\begin{array}{lcll} X + Y & = & [\underline{X} + \underline{Y}, \overline{X} + \overline{Y}] & \textit{Addition} \\ X - Y & = & [\underline{X} - \overline{Y}, \overline{X} - \underline{Y}] & \textit{Subtraction} \\ X \times Y & = & [a, b] & \textit{Multiplication} \\ a & = & \min(\underline{X} \times \underline{Y}, \underline{X} \times \overline{Y}, \overline{X} \times \underline{Y}, \overline{X} \times \overline{Y}) & \\ b & = & \max(\underline{X} \times \underline{Y}, \underline{X} \times \overline{Y}, \overline{X} \times \underline{Y}, \overline{X} \times \overline{Y}) & \\ 1/Y & = & [1/\overline{Y}, 1/\underline{Y}] & \textit{Division} \\ X/Y & = & X \times 1/Y. & \end{array} \tag{2.54}$$

Both the *commutative* and *associative* laws hold for intervals [42], that is, for $A, B, C \in I(\mathbb{R})$

$$\begin{aligned} A+B &= B+A \\ A\times B &= B\times A \\ (A+B)+C &= A+(B+C) \\ (A\times B)\times C &= A\times(B\times C) \\ A+0 &= 0+A=A \\ A\times 1 &= 1\times A=A. \end{aligned} \tag{2.55}$$

On the other hand, the *distributive* law does not always hold [42], that is

$$A(B+C)\neq AB+AC \tag{2.56}$$

but, *sub-distributivity* holds [18]

$$A(B+C)\subseteq AB+AC. \tag{2.57}$$

Interval arithmetic assumes that all quantities are independent of one another. Ignoring the dependencies between quantities leads to illogical results. For example, suppose $X=[0,1]$ then using interval arithmetic $X-X$ returns $[-1,1]$ instead of $[0,0]$. The overestimation of the range in this manner is referred to as the *dependency problem*. It is not difficult to deduce that this problem is likely to affect most interval calculations except those involving expressions where each variable occurs only once, such as $X_1+X_2+X_3+...$ [43].

For most practical purposes, interval arithmetic operations are carried out using finite precision floating point arithmetic, which is referred to as *floating point interval arithmetic* [18]. To ensure the containment property, both upward and downward rounding modes must be exercised in order to account for rounding errors. In this thesis, no distinction is made between real and floating point interval arithmetic unless stated otherwise.

Definition 2.15. (Floating Point Interval Arithmetic) Interval addition, subtraction, multiplication, and division is implemented in floating point arithmetic as follows

$$
\begin{array}{lcll}
X+Y & = & [\oplus_\nabla(\underline{X},\underline{Y}), \oplus_\Delta(\overline{X},\overline{Y})] & \textit{Addition} \\
X-Y & = & [\ominus_\nabla(\underline{X},\overline{Y}), \oplus_\Delta(\overline{X},\underline{Y})] & \textit{Subtraction} \\
X \times Y & = & [a,b] & \textit{Multiplication} \\
a & = & \min(\otimes_\nabla(\underline{X},\underline{Y}), \otimes_\nabla(\underline{X},\overline{Y}), \otimes_\nabla(\overline{X},\underline{Y}), \otimes_\nabla(\overline{X},\overline{Y})) & \\
b & = & \max(\otimes_\Delta(\underline{X},\underline{Y}), \otimes_\Delta(\underline{X},\overline{Y}), \otimes_\Delta(\overline{X},\underline{Y}), \otimes_\Delta(\overline{X},\overline{Y})) & \\
1/Y & = & [\oslash_\nabla(1,\overline{Y}), \oslash_\nabla(1,\underline{Y})] & \textit{Division} \\
X/Y & = & X \times 1/Y &
\end{array}
\tag{2.58}
$$

where $\oplus$, $\ominus$, $\otimes$ denote floating point addition, subtraction, and multiplication, respectively; while the subscripts Δ and ∇ refer respectively to upward and downward rounding modes of floating point arithmetic. It is clear that without specific hardware support, floating point interval arithmetic requires more instructions and storage to carry out than floating point arithmetic.

2.4.2 Interval Functions

Definition 2.16. (Interval Hull) Given a real function $\mathrm{f} : \mathbb{R}^n \to \mathbb{R}^m$ defined in the domain $S \subseteq I(\mathbb{R}^n)$, then the *interval hull* [66] of f in S is the smallest interval vector in $I(\mathbb{R}^n)$ that contains the range of f in S. We denote the interval hull of f in S as $\mathrm{f}^{\mathrm{h}}(\mathbf{S})$.

An interval hull is an example of an *interval function*.

Definition 2.17. (Interval Function) An interval function is a function which takes an interval input and returns an interval output [18].

Definition 2.18. (Interval Extension) Suppose $F : I(\mathbb{R}^m) \to I(\mathbb{R}^n)$ is an interval function defined in $S \subseteq I(\mathbb{R}^n)$, we can say that F is an *interval extension* [18] of the real function $\mathrm{f} : \mathbb{R}^m \to \mathbb{R}^n$ if and only if the following containment property holds

$$F(X) \supseteq \{f(x) \mid x \in X\} \tag{2.59}$$

where $X \subseteq S$ and $x \in \mathbb{R}^n$.

In other words, the interval extension of f contains the range of f over the domain spanned by X. The interval hull is the best possible interval extension, as it provides the sharpest possible estimate of the range of f.

$$f^h(X) \subseteq F(X), \quad \forall F. \tag{2.60}$$

The problem of finding the exact interval hull is generally NP-complete as it is equivalent to optimizing f with respect to the domain spanned by X [66]. Thus, in practice, it is often only possible to compute an interval extension that provides an *enclosure* of the interval hull [66].

Example 2.19. Supposing that $f(x)$ represents a real function, then $F(X)$ is a valid interval extension of $f(x)$ in the following cases

$$\begin{aligned} f(x) = \exp(x) \quad &\rightarrow \quad F(X) = [\exp(\underline{X}), \exp(\overline{X})] \\ f(x) = \cos(x) \quad &\rightarrow \quad F(X) = [-1, 1] \\ f(x) = -\log(x) \quad &\rightarrow \quad F(X) = [-\log(\overline{X}), -log(\underline{X})] \\ f(x) = x(2x-1) \quad &\rightarrow \quad F(X) = X(2X-1) \end{aligned}$$

2.4.3 Interval Extensions of Rational Functions

Definition 2.20. (Rational Functions) We define *rational functions* as functions computable as an algorithm or computer program consisting entirely of the four elementary arithmetic operations $\{+, -, \times, \div\}$, and evaluations of standard functions.

Definition 2.21. (Natural Interval Extension) There is a relatively straight forward method for formulating an interval extension of a rational function: first we write the mathematical expression representing the rational function, then in that expression we

1. Replace each variable by a corresponding interval variable.
2. Replace each arithmetic operation by a corresponding real interval arithmetic operation.
3. Replace each standard function by an interval extension representing its interval hull.

Interval extensions formulated in this way are referred to as *natural interval extensions* [66].

Remark 2.22. It should be noted that it is still possible to formulate an interval extension using a relaxed version of Definition 2.21 [66]. Unless stated otherwise, we will still refer to a *natural interval extension* even when: i) floating point interval arithmetic is used instead of real interval arithmetic (in item 2), and/or ii) a general interval extension is used instead of one that represents the exact interval hull (in item 3).

For example, we can formulate the natural interval extension of $f(x) = x + \exp(x) + xy - xyz$ by i) replacing x, y, and z by interval variables X, Y, and Z, ii) replacing each addition, subtraction, and multiplication operation by interval addition, subtraction, and multiplication operations, and iii) replacing $\exp(x)$ by its interval extension $[\exp(\underline{X}), \exp(\overline{X})]$.

Furthermore, it should also be noted that an interval extension of Rump's example (Equation 2.46) computed using floating point interval arithmetic produces a wide interval that contains the exact result and the numerical instability caused by cancellation, thereby exposing the latter issue [91].

2.4.4 Properties of Interval Extensions

The asymptotic properties of interval extensions as the input variables approach degeneracy ($w(X) \to 0$) can be characterized as either first-order, second-order, third-order, fourth-order, respectively.

Definition 2.23. **(α-order Interval Extension)** Let F denote an interval extension of $f : \mathbb{R}^n \to \mathbb{R}^m$ evaluated over $X \in I(\mathbb{R}^n)$. If there is a constant $K > 0$, independent of X, such that

$$w(F(X)) - w(f^h(X)) \leq K w(X)^\alpha \tag{2.61}$$

for all X with sufficiently small $w(X)$, and a fixed $\alpha > 0$. Then we say that F is an order α interval extension of f, i.e. first order, second order, etc. From [66].

Theorem 2.24. *Natural interval extensions as defined in Definition 2.21 are first-order interval extensions [66].*

Higher order interval extensions are more desirable as they have better convergence properties, which translates in to tighter interval bounds, especially when $w(X)$ is small. Higher order interval extensions can be derived, for example, using the *Taylor model* method [95], which is roughly equivalent to taking the natural interval extension of a Taylor series expansion of the original function.

It should be noted that the α-order is only an approximation of actual asymptotic behaviour in i) interval extensions computed using floating point interval arithmetic, or ii) natural interval extensions formulated under the relaxed definition (see Remark 2.22). It does not take into account factors such as rounding error in floating point computation, or the precision in which the range of standard functions are evaluated. Therefore, the α-order cannot be considered as valid

especially when the width of the interval variables are of a similar order of magnitude to the machine precision.

Whatever the order of an interval extension, we tend to get tighter bounds as the width of the interval input decreases. Therefore, one common way to calculate tighter bounds is to sub-divide the interval input X into n sub-intervals

$$X = \bigcup_{i=1}^{n} X_i$$

and then compute the union of the results of each sub-interval [42],

$$f(X) = \bigcup_{i=1}^{n} f(X_i)$$

However, this can be very costly especially in the multi-dimensional case.

Definition 2.25. (Inclusion Isotonicity) An interval function F is said to be *inclusion isotonic* or *inclusion monotonic* provided $X \subset Y$ and $X, Y \in I(\mathbb{R}^n)$ implies $F(X) \subset F(Y)$ [66].

Theorem 2.26. *Natural interval extensions as defined in Definition 2.21 are inclusion isotonic.*

2.5 Optimization

Multivariate optimization of some form is required in almost all disciplines of science, engineering, mathematics, finance, and management. This thesis is primarily concerned with finding the solutions to constrained optimization problems of the form

$$\begin{aligned} P: \quad & z = \min \quad f(x) \\ & \text{s.t.} \quad\;\; x \in S \\ & \qquad\quad\;\; S = \{x \mid x_i^l \leq x_i \leq x_i^u,\ g_j(x) \leq 0,\ h_k(x) = 0\} \\ & \qquad\quad\;\; \forall i \in \{1, 2, .., n\},\ \forall j \in \{1, 2, .., q\},\ k \in \{1, 2, 3, .., r\} \end{aligned} \tag{2.62}$$

where P is optimized with respect to the set of *optimization variables* x; x^l and x^u are the set of lower and upper *variable bound constraints*, respectively; $f(x)$ is the *objective function*; $g_j(x) \leq 0$ and $h_k(x) = 0$ are the sets of *inequality constraints* and *equality constraints* respectively; and S is the *feasible set* defined by the abovementioned constraints.

This section provides a very brief introduction to optimization, touching on some basic definitions, then discussing convex analysis, and conditions for optimality - and ending with a brief overview of different local optimization methods.

2.5.1 Local, Global, and ϵ-Global Minima

In optimization local, global, and ϵ-global minima are defined as follows:

Definition 2.27. (Local Minimum) Suppose that $x^* \in S$ and there exists a constant $\epsilon > 0$ such that

$$f(x) \geq f(x^*) \quad \forall x \in S, \ s.t. \ \|x - x^*\| < \epsilon$$

then $f(x^*)$ is a *local minimum* and x^* is a *local minimizer.*

Definition 2.28. (Global Minimum) Suppose that $x^* \in S$ is a feasible solution and

$$f(x) \geq f(x^*) \quad \forall x \in S,$$

then $f(x^*)$ is a *global minimum* and x^* is a *global minimizer.*

Definition 2.29. (ϵ-Global Minimum) Suppose that $x^* \in S$ is a feasible solution, $\epsilon \geq 0$ is a small predetermined tolerance, and

$$f(x) \geq f(x^*) - \epsilon \quad \forall x \in S,$$

then $f(x^*)$ is a *ϵ-global minimum* and x^* is a *ϵ-global minimizer.* A solution is said to be *ϵ-optimal* if it yields an *ϵ-global minimum.* [26].

Definition 2.30. (ϵ-Feasibility) Suppose that $\epsilon \geq 0$ is a small predetermined tolerance, and

$$\|x^* - x\|_\infty < \epsilon \quad \exists x \in S,$$

then x^* is a *ϵ-feasible* solution with respect to S.

In the same regard, P is an *ϵ-feasible* problem if such an *ϵ-feasible* solution exists, otherwise it is an *ϵ-infeasible* problem.

2.5.2 Categories of Optimization Problems

Optimization problems can be categorized according to the type of objective function, constraints, and variables they incorporate, and the type of solution that is required [83]. Some of the major categories encountered in practice are as follows

- Categories with respect to optimization variable type
  - **Continuous Optimization**. The feasible region S is a subset of a Euclidean space.
  - **Integer Optimization**. The feasible region S is finite or countable.
  - **Mixed-Integer Optimization**. $S = U \times V$, where U is a subset of a Euclidean space, and V is finite or countable.
- Categories with respect to objective function and constraints. Note that variable bound constraints are not considered as constraints in the following definitions.
  - **Unconstrained Optimization**. There are no constraints to be satisfied.
  - **Constrained Optimization**. The feasible region S is defined by a set of constraints to be satisfied.
  - **Linear Optimization**. The functions $f(x)$, $\{g_j(x)\}_{j=1}^{q}$, and $\{h_k(x)\}_{k=1}^{r}$ are linear functions of x.
  - **Non-Linear Optimization**. The functions $f(x)$, $\{g_j(x)\}_{j=1}^{q}$, and $\{h_k(x)\}_{k=1}^{r}$ **may** be non-linear functions of x.
  - **Convex Optimization**. The function $f(x)$ and the feasible region S are convex.
  - **Non-Convex Optimization**. The function $f(x)$ and the feasible region S **may** be non-convex.
- Categories with respect to the type of solution required
  - **Local Optimization** A problem where a local minimum is sought (see Definition 2.27).
  - **Global Optimization** A problem where the global minimum is sought (see Definition 2.28).

It is clear that most problems are likely to fall under multiple categories. Although integer optimization or integer programming will be mentioned in later chapters, this thesis primarily concerns the solving of continuous, constrained, non-linear, non-convex, global optimization problems within the realm of quantum chemistry.

Definition 2.31. (Non-Linear Programming) A (continuous) non-linear programming (NLP) problem can be expressed as

$$
\begin{array}{lllll}
NLP: & z = \min & f(x) & & \\
& \text{s.t.} & g_j(x) \leq 0 & \forall j \in \{1, 2, 3, .., q\} & \\
& & h_k(x) = 0 & \forall k \in \{1, 2, 3, .., r\} & \\
& & x_i^l \leq x_i \leq x_i^u & \forall i \in \{1, 2, 3, .., n\} &
\end{array} \tag{2.63}
$$

where $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$, $g_j : \mathbb{R}^n \to \mathbb{R}$, $h_k : \mathbb{R}^n \to \mathbb{R}$, $x_i \in \mathbb{R}$, $x_i^l \in \mathbb{R}$, $x_i^u \in \mathbb{R}$; and $f(x)$, $g_j(x)$, $h_k(x)$ may be non-linear functions.

2.5.3 Convexity

This section defines convex sets, convex functions, and convex problems; and details various properties.

Definition 2.32. (Convex Sets) We say the set $S \subseteq \mathbb{R}^n$ is *convex* if, for all $x, y \in S$ and for all $\lambda \in [0, 1]$ the line segment $\lambda x + (1 - \lambda)y$ is also in S

Examples of convex sets include

- A Line
- Open and closed half-spaces
- A Hypercube
- All points inside and on a circle
- All points inside and on a polygon

Supposing that $S_1 \subseteq \mathbb{R}^n$ and $S_2 \subseteq \mathbb{R}^n$ are convex sets, then the following are also convex sets

1. the intersection $S_1 \cap S_2$
2. the sum $S_1 + S_2$
3. the scalar product λS_1 where $\lambda \in \mathbb{R}$

Definition 2.33. (Convex Hull) Let $S \subseteq \mathbb{R}^n$, then the convex hull of S, denoted as $H(S)$, is defined as the intersection of all the convex subsets of $\mathbb{R}^n$ containing S as a subset.

In other words, $H(S)$ is the tightest convex enclosure of S.

Definition 2.34. (Convex Functions) Let $S \subseteq \mathbb{R}^n$ be a convex subset, and $f : \mathbb{R}^n \to \mathbb{R}$ be a function defined in S. The function $f(x)$ is said to be *convex* if and only if: for all $x, y \in S$ and for all $\lambda \in [0, 1]$,

$$f(\lambda x + (1 - \lambda)y) \leq \lambda f(x) + (1 - \lambda)f(y). \tag{2.64}$$

Conversely, a function is *non-convex* if it is not *convex*, while a function f is said to be *concave* if $-f$ is convex.

Examples of convex functions include

- $a_1x_1 + a_2x_2 + .. + a_nx_n$
- $\exp x$
- $ax^2 + bx + c$, where $a \geq 0$
- $\sin(x)$ for $x \in [\pi, 2\pi]$

Two or more convex functions can be combined to form new convex functions in a number of different ways. Let $f_i(x), \forall i \in \{1, 2, .., n\}$ be convex functions defined in a convex subset $S \subseteq \mathbb{R}^n$, then the following functions are also convex in S:

- The summation $f(x) = \sum_{i=1}^{n} f_i(x)$
- The scalar product $f(x) = \lambda f_i(x), \forall i \in \{1, 2, .., n\}$, where $\lambda \in \mathbb{R}^+$ is positive.
- The composite function $g(f_i(x)), \forall i \in \{1, 2, .., n\}$, providing that $g(x)$ is a non-decreasing convex function defined in the range of $f_i(x)$.
- The maximum function $f(x) = \max\{f_1(x), .., f_n(x)\}$, providing that $f_i(x), \forall i \in \{1, 2, .., n\}$ is bounded from above.
- The (negation of the) geometric mean, $f(x) = -(\prod_{i=1}^{n} f_i(x))^{1/n}$, $x \in \mathbb{R}^{n+}$ [26]

Theorem 2.35. *If the set $S \subseteq \mathbb{R}^n$ is convex and $f : \mathbb{R}^n \to \mathbb{R}$ is twice differentiable on S, then f is convex in S if and only if its Hessian matrix $H_f = f''(x)$ is positive semi-definite for all $x \in S$.*

Definition 2.36. (Convex Constraints) If the set $S \subseteq \mathbb{R}^n$ is convex and $x \in S$, then

- $g(x) \leq 0$ is a convex constraint if and only if g is a convex function in S.
- $h(x) = 0$ is a convex constraint if and only if h is a linear function in S.

Definition 2.37. (Convex Problems) We can say that P (in Equation 2.62) is a *convex optimization problem* if and only if

1. $f(x)$ is a convex function in S,
2. $g_i(x)$ is a convex function in S, $\forall i \in 1, 2, ..p$, and
3. $h_j(x)$ is a linear function in S, $\forall j \in 1, 2, ..q$

Definition 2.38. (Linear Programming) The Linear Programming (LP) Problem is a well known instance of a convex problem

$$\begin{array}{lll} LP: \quad z = \min & c_1x_1 + c_2x_2 + .. + c_nx_n & \\ \text{s.t.} & a_{11}x_1 + a_{12}x_2 + .. + a_{1n}x_n & \leq b_1 \\ & a_{21}x_1 + a_{22}x_2 + .. + a_{2n}x_n & \leq b_2 \\ & & . \\ & & . \\ & & . \\ & a_{m1}x_1 + a_{m2}x_2 + .. + a_{mn}x_n & \leq b_m \\ & x_k \geq 0 & \forall k \in \{1, 2, 3, .., n\} \end{array} \tag{2.65}$$

where $c \in \mathbb{R}^n$, $a \in \mathbb{R}^{m \times n}$, and $x \in \mathbb{R}^n$.

Theorem 2.39. *If the set $S \subset \mathbb{R}^n$ is bounded and convex and if* f *is convex in S, then* f *has a unique global minimizer in S.*

This theorem implies that a local minimum of a convex function is also its global minimum [45]. A similar theorem applies to convex problems.

Theorem 2.40. *A convex problem has a unique global minimizer.*

This theorem implies that a local optimization algorithm can be used to reliably obtain the global minimum of a convex problem.

Definition 2.41. (Convex Relaxation of a Function) Suppose f(x) is a continuous function defined in the convex set $S \subseteq \mathbb{R}^n$. Then $\underline{\text{f(x)}}$ is a *convex relaxation* of f(x) in S, if

1. $\underline{\text{f(x)}}$ is a convex function in S, and
2. $\underline{\text{f(x)}} \leq \text{f(x)}, \forall x \in S$

A *linear relaxation* is a special instance of a convex relaxation where $\underline{\text{f(x)}}$ is linear.

Definition 2.42. (Convex Relaxation of a Problem) Let P and P' be constrained optimization problems of the form specified in Equation 2.62; and $\mathrm{f}(\mathrm{x})$ and $\underline{\mathrm{f}(\mathrm{x})}$ be the objective functions of P and P', respectively. Then P' is a *convex relaxation* of P, if

1. P' is a convex problem, and
2. $\underline{\mathrm{f}(\mathrm{x})} \leq \mathrm{f}(\mathrm{x}), \forall x \in S$

Convex relaxations are often used by optimization algorithms as a means for obtaining a lower bound of the objective function across a given region. There are many ways in which a convex relaxation can be constructed for a given problem, finding the best one is challenging. The ideal convex relaxation is one that gives the tightest possible bound on the objective function of the primal problem.

2.5.4 Necessary and Sufficient Conditions for Local Minimum

i) Unconstrained Optimization

For the following theorems we assume an unconstrained problem of the form:

$$x^* = \operatorname{argmin} f(x), \text{ where } f : \mathbb{R}^n \to \mathbb{R}, \ x \in \mathbb{R}^n.$$

In unconstrained optimization, the necessary and sufficient conditions for a local minimum can be derived from the gradient and the Hessian of the objective function.

Theorem 2.43. ***(Necessary Conditions)*** *If x^* is a local minimizer of f, then $f'(x^*) = 0$.*

Definition 2.44. (Stationary point) x^* is a *stationary point* of f, if and only if $\mathrm{f}'(\mathrm{x}^*) = \mathbf{0}$

Since a stationary point may correspond to either a local minima or a saddle point, the necessary condition is not sufficient to guarantee a local minimum.

Theorem 2.45. ***(Sufficient Conditions)*** *x^* is a local minimizer of f if and only if,*

1. *x^* is a stationary point of f, and*
2. *The Hessian matrix of f, $\mathrm{f}''(\mathrm{x}^*)$, is positive definite.*

ii) Constrained Optimization

The Lagrange function and its derivatives are important in expressing theoretical conditions of local optimality in constrained optimization.

Definition 2.46. Lagrange's Function The Lagrangian function or Lagrangian dual with respect to P (Equation 2.62) is defined as

$$L(x, \lambda, \mu) = f(x) + \sum_{j=1}^{q} \lambda_j g_j(x) + \sum_{k=1}^{r} \mu_k h_k(x) \tag{2.66}$$

where $\lambda \in \mathbb{R}^q$ and $\mu \in \mathbb{R}^r$ are referred to as *Lagrange Multipliers*.

The derivative of the Lagrangian function with respect to x can be written as follows

$$L'_x(x, \lambda, \mu) = \nabla f(x) + \sum_{j=1}^{q} \nabla \lambda_j g_j(x) + \sum_{k=1}^{r} \nabla \mu_k h_k(x). \tag{2.67}$$

The following theorems by Karush [63], and Kuhn and Tucker [76] (KKT) establishes the necessary and sufficient conditions for a local minimum in a constrained optimization problem.

Theorem 2.47. ***(First-Order KKT Necessary Conditions)*** *If* x^* *is a local minimizer of* P*, then there exist Lagrangian multipliers* λ *and* μ *such that*

1. $L'_x(x^*, \lambda^*, \mu^*) = 0$ ***(Optimality)***
2. $\lambda \geq 0$ ***(Dual Feasibility)***
3. $\lambda_j g_j(x^*) = 0, \forall j \in \{1, 2, .., q\}$ ***(Complementary Slackness)***
4. $g_j(x^*) \leq 0, \forall j \in \{1, 2, .., q\}$ ***(Primal Feasibility)***
5. $h_k(x^*) = 0, \forall k \in \{1, 2, .., r\}$ ***(Primal Feasibility)***

(Note that, unlike λ, μ is not restricted in sign)

Theorem 2.48. ***(Second-Order KKT Sufficient Conditions)*** *Supposing that* $\mathrm{f}, \mathrm{g}, \mathrm{h}$ *are twice-differentiable, and*

1. *The first-order KKT conditions in Theorem 2.47 are satisfied, and*

2. *The Hessian matrix of Lagrange's function,* $\mathbf{W}^* = L''_{xx}(x^*, \lambda, \mu)$ *is positive definite in a subspace of* $\mathbb{R}^n$ *defined by*

   - $\mathbf{y}^T \mathbf{W}^* \mathbf{y} > 0$, *where* $\mathbf{y}$ *is a feasible active direction such that*
   - $\mathbf{y}^T \nabla g_j(x^*) = 0$ *for* j *where* $\lambda_j > 0$
   - $\mathbf{y}^T \nabla h_k(x^*) = 0$, $\forall k \in \{1, 2, .., r\}$

then $\mathbf{x}^*$ *is a local minimizer of* P.

2.5.5 Local Optimization Methods

The most primitive approaches for solving optimization problems are *search techniques*. They involve partitioning the feasible domain of the problem into a series of grid points, whereby the point corresponding to the smallest objective value is declared to be the minimum. This approach works reasonably well for one dimensional problems, but does not scale towards multiple dimensions due to exponential complexity.

More sophisticated optimization methods have since been developed, mostly during the past fifty years, spurred along by the widespread use of electronic computers. The majority of these methods are aimed at finding local minima.

Local optimization methods are mainly greedy algorithms designed to find the stationary point nearest to the specified starting point. As such, these methods often do not discriminate whether the final result is locally optimal, globally optimal, or even a saddle point. Different approaches are used for unconstrained and constrained optimization.

Unconstrained Problems

The local minimum of an unconstrained optimization problems are typically based on *iterative methods* that *descend* towards a stationary point from an initial starting point (Algorithm 2). In a successful optimization, the starting point and subsequent iterations form a converging sequence, $\{x_0, x_1, ..., x_k, .., x^*\}$, towards the local minimum where, $x_i \in \mathbb{R}^n$, $\forall i = \{0, 1, 2, ..\}$.

Definition 2.49. (Descent Property) Using the above notation, the *descent property* that holds between each successive iteration can be expressed as

$$\mathbf{f}(\mathbf{x_{k+1}}) \leq \mathbf{f}(\mathbf{x_k})$$

The update procedure of an iterative method can typically be written as

$$\mathbf{x_{k+1}} = \mathbf{x_k} + \lambda_k \mathbf{p_k}$$

where $\mathbf{p_k} \in \mathbb{R}^n$ is the *descent direction*, and $\lambda_k \in \mathbb{R}$ is the *step size* along the descent direction. The efficiency of an iterative method is highly dependent on how $\mathbf{p_k}$ is determined - one obvious approach is to follow the direction of *steepest descent*, that corresponds to the *negative* of the gradient of f at $\mathbf{x_k}$. More superior descent methods than steepest descent include the *Conjugate Gradient Method* [25], the *Newton's method*, and the *Quasi-Newton method* [16].

The step size is either set to $\lambda_k = 1$, or determined based on a *line search* along the descent direction [45]. In the latter case, λ_k corresponds to the step size that exactly or approximately minimizes the objective function.

Algorithm 2 A generic iterative descent method.

Supply an initial guess $\mathbf{x_0}$
Initialize $k = -1$
while Not Converged **do**
  Set $k = k + 1$
  Calculate a descent direction $\mathbf{p_k}$
  Determine an appropriate step size λ_k by line-search, or let $\lambda_k = 1$
  Set $\mathbf{x_{k+1}} = \mathbf{x_k} + \lambda_k \mathbf{p_k}$
end while
If the procedure converges then let $\mathbf{x}^* = \mathbf{x_{k+1}}$

The convergence criteria of the iterative method should ensure that the distance between $\mathbf{x_{k+1}}$ and $\mathbf{x}^*$ is below a certain threshold ϵ. However, since $\mathbf{x}^*$ is not known, a surrogate criteria must be used. For example,

$$\|\mathbf{x_{k+1}} - \mathbf{x_k}\| < \epsilon$$

or

$$\|\mathbf{f}(\mathbf{x_{k+1}}) - \mathbf{f}(\mathbf{x_k})\| < \epsilon$$

The convergence criteria can also be based on the necessary or sufficient conditions for optimality.

Constrained Problems

Penalty (or *Barrier*) methods are a widely used class of approaches for solving non-linear constrained optimization problems [94]. The main idea is to construct

a unconstrained *dual problem* where the constraints of the primal problem are embedded in to its objective function, so that there is a penalty on the new objective if any of the constraints are not satisfied [94].

For a problem of the form specified in Equation 2.62 - a simple dual problem, denoted as φ, can be expressed as

$$\mathbf{x}_{\varphi} = \text{argmin}_{x}\varphi(\mathbf{x}, \mu, \lambda) \tag{2.68}$$

and

$$\varphi(\mathbf{x}, \mu, \lambda) = f(x) + \sum_{j=1}^{q} \mu_j \max[0, g_j(x)] + \sum_{k=1}^{r} \lambda_k |h_j(x)| \tag{2.69}$$

where $\mu_i \in \mathbb{R}^+, \forall j \in \{1, 2, .., q\}$ and $\lambda_j \in \mathbb{R}^+, \forall j \in \{1, 2, .., q\}$ are fixed *penalty terms*. When both the equality and inequality constraints are satisfied, it is clear that the *penalty function* component of the objective function

$$\sum_{j=1}^{q} \mu_j \max[0, g_j(x)] + \sum_{k=1}^{r} \lambda_k |h_j(x)|$$

equals zero. However, to guarantee the desired convergence of $\mathbf{x}_{\varphi} \rightarrow \mathbf{x}^*$, the values of μ and λ must approach infinity, which for this particular penalty function leads to an ill-conditioned problem [94]. Another problem with the dual function above is that it is highly non-smooth, making it difficult for a standard iterative descent method to find a solution.

The most commonly used penalty functions are based on logarithms which are smoother, but still suffer from the problem of ill-conditioning as μ and λ approaches infinity. The latter problem can be avoiding by instead using the *Augmented Lagrangian Method* [117], which works by incorporating the Lagrange multipliers within the dual problem.

The theoretical and computational aspect of local optimization have seen extensive development in recent decades. On the other hand, methods in global optimization have received comparatively less attention, primarily due to their relative intractability. For practical reasons, there has long been a tendency in many fields of research to apply local optimization to address what are essentially global optimization problems. This tendency still persists today despite the issues known to be associated with it. Most notably, for non-convex problems, local optimization is not guaranteed to yield a global minimum.

Part

II

Interval Analysis Approaches

In the following two chapters interval analysis techniques are used to address the research objective: *to place rigorous error bounds on all numerical errors associated with a Hartree-Fock computation*.

Chapter 3 introduces methods for placing rigorous interval bounds on significant Hartree-Fock quantities such as the total energy, the molecular orbital energies, the molecular orbital coefficients, and derived electronic properties. A program that is able to perform these calculations for arbitrary closed-shell systems is also introduced.

Chapter 4 uses this program to investigate the impact of numerical errors in Hartree-Fock computations. In particular, it focuses on a number of input and design related factors which are relevant to Hartree-Fock computations, and the implementation of Hartree-Fock codes.

Chapter

3

Development of the Interval Hartree-Fock Program

Contents

3.1 Introduction

During the past twenty years the vast majority of computational chemistry programs have been written to use IEEE 754 double precision floating point arithmetic [55]. This was motivated both by the perception that double precision was required in order to obtain sufficiently accurate results and the fact that efficient implementations of this standard were available on all commonly used high performance microprocessors. With a few notable exceptions [162, 120], relatively little attention was given to using alternative levels of arithmetic precision. In the last few years this situation has changed dramatically. This has been largely prompted by the move by manufacturers of graphics and gaming hardware into the general purpose computing market, the ability of this new hardware to perform floating point computations at a rate that was significantly higher than that of contemporary general purpose microprocessors, and the fact that this was (initially) only possible when using single precision arithmetic with rounding modes that were not entirely IEEE 754 compliant. Two high profile early examples are the Sony Toshiba IBM (STI) Cell Broadband Engine [62] that forms the basis for the PlayStation 3 and the NVIDIA GTX8800 CUDA based graphics card [164].

More recently initiatives in exascale computing, green computing, and many-core architectures have provided added impetus for re-considering the role of arithmetic precision in computational chemistry codes. In these cases the sheer number of operations performed, the energy required to perform the operations, and the available bandwidth for moving data on and off the processor chip are all causes for wanting to represent data with the minimal number of bytes possible. Also, developments in field programmable gate array (FPGA) technology mean that computational devices can now be constructed on the fly to use arbitrary levels of precision that may not be standard single or double precision.

All of the above considerations motivate the development of new tools to study the impact of numerical precision and various other underlying numerical approximations on the quality of Hartree-Fock computation. In pursuit of this goal, this chapter outlines the development of an *Interval Hartree-Fock* program, which is designed to calculate rigorous interval bounds on numerical errors in quantities such as the total energy, atomic orbital energies, molecular orbital coefficients, and derived electronic properties.

Interval Hartree-Fock uses interval analysis techniques to calculate error bounds on quantities generated at each step of a Hartree-Fock calculation, which leads to an error bound on the final result. The errors associated with arithmetic precision, including rounding errors, are automatically bounded and propagated

by simple interval arithmetic. However, additional work is required to bound truncation errors due to numerical approximation and errors propagated as a result of mathematical transformation. This chapter specifically discusses the following topics in this regard:

- Methods for bounding truncation errors in the the one and two-electron integral construction step; focusing in particular on the errors introduced by the numerical approximation of $F_m(T)$,
- The diagonalization of Fock matrices which contain uncertain elements. Each element is represented by an interval bounding numerical errors from previous steps in the calculation.
- Methods for tightening error bounds by applying compensated summation and *cascaded accumulated summation* to reduce the accumulation of rounding errors in numerically intensive sections of the interval code.

The chapter is structured with the following sections: Section 3.2 provides a brief introduction to techniques for accurately evaluating summation and dot products of floating point and interval quantities; Section 3.3 outlines approaches for bounding numerical errors in $F_m(T)$; Section 3.4 discusses the calculation of error bounds on the total energy; Section 3.5 describes approaches for diagonalizing interval Fock matrices; and Section 3.7 provides some conclusions and outlines possible future work.

3.2 Accurate Summation and Dot Products

This section briefly introduces methods for accurately evaluating summation and dot products in both floating point and interval arithmetic. These methods will be applied towards sections of the Hartree-Fock code where numerical errors are likely to accumulate - in order to calculate more accurate results in floating point arithmetic, and sharper error bounds in interval arithmetic.

The majority of scientific codes have components that involve summation

$$s = \sum_{i=1}^{n} x_i \tag{3.1}$$

or dot products

$$d = \mathbf{x} \bullet \mathbf{y} = \sum_{i=1}^{n} x_i y_i \tag{3.2}$$

where $\{x_i\}_{i=1}^n, \{y_i\}_{i=1}^n \in \mathbb{R}^n$. When these operations involve a large number of terms, rounding errors tend to accumulate. It is therefore pertinent to consider how best to code summation and dot product operations in order to minimize rounding errors. A naive implementation of summation is *recursive summation*

$$\begin{aligned} s &= s_1 \\ s_i &= x_i + s_{i+1}, \quad 0 \leq i \leq n \end{aligned} \tag{3.3}$$

expressed iteratively in Algorithm 3. Note that we use the directed rounding notations from Section 2.4.

Algorithm 3 RecursiveSummation(x)

```
1: s = 0
2: for all i = 1 to n do
3:     s = ⊕∘(x_i,s)
4: end for
5: return s
```

Since the rounding errors resulting from each addition operation is repeatedly accrued in the sub-total s, recursive summation is not particularly accurate.

More accurate approaches for implementing summation have been developed. In this section we introduce two such approaches in *compensated summation* originally due to Kahan [61] and *cascaded accumulated summation* originally due to Wolfe [161], both of which can also be generalized to perform accurate dot products [116].

3.2.1 Error Free Transformation

Both the compensated and cascaded accumulated summation methods rely on another method for estimating the rounding error resulting from a single arithmetic operation. An often used method is the *error-free transformation* [116], which allows the rounding error of floating point addition and multiplication to be calculated exactly.

The error free transformation for floating point addition is given in Algorithm 4 [73]. It *transforms* two input floating point values $x, y \in \mathbb{R}$ and returns two output values $s, e \in \mathbb{R}$ where $s = \mathrm{fl}(x + y)$ is the floating point addition of x and y, and e

is the exact rounding error of s. This transformation can be performed entirely in round to nearest arithmetic.

Algorithm 4 EF_Add(x,y)

Ensure: $s + e = x + y$

1: $s = \oplus_\circ(x,y)$
2: z = $\ominus_\circ(s,x)$
3: $e = \oplus_\circ(\ominus_\circ(x,\ominus_\circ(s,\text{z})),\ominus_\circ(y,\text{z}))$
4: **return** s, e

The error free transformation for floating point multiplication is given in Algorithm 5 [21]. It accepts two input values $x, y \in \mathbb{R}$ and returns two output values $p, e \in \mathbb{R}$, where p is the result of the floating point multiplication of x and y, and e is the exact rounding error of p.

Algorithm 5 p, e = EF_Product(x,y)

Ensure: $p + e = x \times y$

1: $p = \otimes_\circ(x,y)$
2: $[x_1,x_2]$ = Split(x)
3: $[y_1,y_2]$ = Split(y)
4: $e = \otimes_\circ(x_2, y_2) - \ominus_\circ(\ominus_\circ(\ominus_\circ(p,\otimes_\circ(x_1, y_1)),\otimes_\circ(x_2,y_1)),\otimes_\circ(x_1,y_2))$
5: **return** p, e

The Split(.) function given in Algorithm 6 [21, 116], partitions a floating point number x into two parts x_1 and x_2, where each part has at most $s-1$ non-zero bits in its mantissa. For a mantissa consisting of t bits, we define s to be $s = \lceil \frac{t}{2} \rceil$. Hence in IEEE754 double precision we have $t = 53$ and $s = 27$.

Algorithm 6 x_1, x_2 = Split(x)

1: $factor = 2^s + 1$
2: $c = \otimes_\circ(factor, x)$
3: $x_1 = \ominus_\circ(\text{c},\ominus_\circ(c,x))$
4: $x_2 = \ominus_\circ(\text{x}, x_1)$

3.2.2 Compensated Summation

The idea behind compensated summation is simple: to correct the result of a summation using a carefully constructed series of *error correcting terms*. There are multitudes of ways in which this can be implemented. In this work, we use the error free transformation to provide error correcting terms, and apply the correction at each iteration.

Algorithm 7 s = CompensatedSummation(x)

1: $s = x_1$
2: $e = 0$
3: **for all** i = 2 to n **do**
4: $\quad y = \oplus_\circ(x_i, e)$
5: $\quad s, e$ = EF_Add(s,y)
6: **end for**
7: **return** s

An implementation of compensated summation is given in Algorithm 7. The value of e generated at each iteration is added on to the next term in the summation, x_{i+1}. If the magnitude of s is greater than or equal to the magnitude of x_{i+1} then e has the effect of correcting s at each iteration. However, if x_{i+1} is much greater than s, then e is likely to be rounded off when added to x_{i+1}.

There are other alternatives to this scheme with their own set of merits, for instance a running total of the errors from each iteration can be kept, and then added to s at the end.

It is also possible to implement compensated summation for interval summation as proposed by Milthorpe [105], Clarke and Rendell [122] and Rendell et. al. [121]. An outline of this approach is provided in Algorithm 8. The purpose here is not specifically to reduce rounding errors, but to reduce the interval width expanded as a result of rounding errors.

This approach is equivalent to applying compensated summation to the interval endpoints. Unlike the previous algorithm, upward rounding mode must be applied to ensure containment. Furthermore, the lower and upper bound of the total must also be accumulated separately. On the other hand, error free transformation can still be used as before since there is no ambiguity over the results it produces.

It should be noted that Doubly Compensated Summation [118] refers to a more accurate variation of compensated summation where a correction term is also calculated for the addition of e on to x_{i+1}.

Compensated summation can also be extended to perform dot products. In this variation of the summation scheme, correction terms must also be calculated for the multiplication operations performed in each iteration. The implementation given in Algorithm 9 is due to Ogita, Rump, and Oishi [116]. It can be described as follows: e_p is the rounding error from the product of x_i and y_i; e_s is the rounding error from the addition of y and s; e_p is added to the value of e_s carried over from the previous iteration and then added to p as a corrective term. This approach is

Algorithm 8 s = CompensatedSummation$^I(X)$

1: s_1 = -$\underline{X_1}$
2: s_2 = $\overline{X_1}$
3: $e_1, e_2 = 0$
4: **for all** i = 2 to n **do**
5: $y_1 = \oplus_\triangle(\text{-}\underline{X_i}, e_1)$
6: $y_2 = \oplus_\triangle(\overline{X_i}, e_2)$
7: s_1, e_1 = EF_Add(s_1,y_1)
8: s_2, e_2 = EF_Add(s_2,y_2)
9: **end for**
10: **return** s = [-s_1, s_2]

based on the assumption that while there is no use in adding e_p to p because of rounding, $e_p + e_s$ together may be large enough to have a corrective effect on p. Interval dot products follows directly from this and interval summation.

Algorithm 9 s = CompensatedDotProduct(x, y)

1: $e_p, e_s = 0$
2: **for all** i = 1 to n **do**
3: p, e_p = EF_Product(x_i, y_i)
4: $t = \oplus_\circ(p, \oplus_\circ(e_s, e_p))$
5: s, e_s = EF_Add(s, t)
6: **end for**
7: **return** s

3.2.3 Cascaded Accumulated Summation

Compensated summation is effective only when the magnitude of the correction term is significant relative to the magnitude of the term that it is applied to. This is a reflection of broader issues involved with summing terms of greatly differing magnitudes, which often leads to a loss of precision in the smaller terms. For an extreme example of this problem, suppose that we are using double precision floating point arithmetic and we want to compute the sum

$$s = (...(((1.0 + \underbrace{1e-17) + 1e-17) + 1e-17) + ...) + 1e-17}_{10^{17} times}. \tag{3.4}$$

The exact value of s is 2.0, however round to nearest arithmetic will produce a result of 1.0. This is because $1e-17$ is smaller than the machine epsilon, and will be rounded out every time it is added to 1.

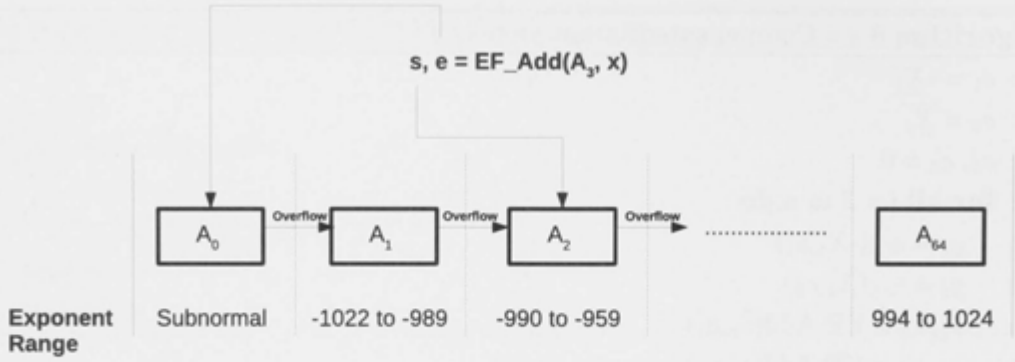


Figure 3.1: An IEEE754 double precision number represented by 64 normal accumulators (A_1 to A_{64}), and 1 sub-normal accumulator (A_0). An example is provided where s is added to A_3, and its residual e to A_0.

The approaches to address this issue often involve pre-sorting the terms by magnitude and then performing the summation. When all the terms are of the same sign, then accumulating the terms in increasing order of magnitude is often the best approach [116], as it gives the smaller terms a chance to accumulate before being added on to the larger terms. For instance, we can rearrange Equation 3.4 so that $1e-17$ is accumulated first, and only adding 1.0 at the end. The main drawback of this approach in the general case is the expense involved in having to sort the terms.

Cascaded Accumulated Summation addresses the abovementioned problem without the need to sort [116]. The idea is to accumulate the final total using a series of accumulators each holding partial sums belonging to an exponent range. When the total of an accumulator exceeds its exponent range it *cascades* on to the next accumulator. Thus, only terms of similar magnitudes are accumulated within the same accumulator. The set of accumulators must therefore span the entire range of exponents plus the sub-normal numbers. For example, to represent IEEE754 double precision, the accumulators must span 2048 unique exponent values [1].

The routine for adding a term x_i on to a set of accumulators $A = \{A_0, A_2, ...A_m\}$ is expressed in Algorithm 10. The function GetExponentIndex(x) determines the index of the accumulator corresponding to the exponent of x. The rounding error resulting from adding x_i to A_k is compensated recursively back on to A. An illustration of the accumulator used in this routine is given in Figure 3.1.

The main cascaded accumulated summation algorithm is expressed in Algorithm 11. After the main loop is performed, it can be proven that the accumulators together represent the exact value of the sum [116]. However, the final result

[1]Note that two of these are reserved for sub-normal numbers and infinity

Algorithm 10 A = CasAccum(A, x)

1: k = GetExponentIndex(x)
2: s, e = EF_Add(x, A_k)
3: $A_k = s$
4: **if** k $\neq$ GetExponentIndex(s) **then**
5: $\quad A_k = 0$
6: $\quad A$ = CasAccum(A, s)
7: **end if**
8: **return** A = CasAccum(A, e)

produced by summing together the accumulators will inevitably be affected by rounding errors. To limit this, the accumulators are usually summed, in order, using either compensated summation or doubly compensated summation [116].

Algorithm 11 s = CascadedSummation(x)

1: $A = \emptyset$
2: **for all** i = 1 to n **do**
3: $\quad A$ = CasAccum(A, x_1)
4: **end for**
5: **return** s = SumAccum(A)

The interval arithmetic equivalent of cascaded accumulated summation was devised in [122, 121], and shown in Algorithm 12. It involves using separate sets of accumulators for evaluating the lower bound $\underline{A} = \{\underline{A}_0, \underline{A}_2, ...\underline{A}_m\}$ and the upper bound $\overline{A} = \{\overline{A}_0, \overline{A}_2, ...\overline{A}_m\}$ of the total sum , together denoted as $A = \{\underline{A}, \overline{A}\}$. The fact that both accumulators are guaranteed to represent the exact result is helpful here since the existing CasAccum routine can be re-used without further modifications.

Algorithm 12 A = CasAccum^I(A, X)

1: $\underline{A}$ = CasAccum($\underline{A}$, $\underline{X}$)
2: $\overline{A}$ = CasAccum($\overline{A}$, $\overline{X}$)
3: **return** $A = \{\underline{A}, \overline{A}\}$

Cascaded accumulated summation can also be generalized towards dot products by incorporating the multiplicative term. A general outline of this is given in Algorithm 13 [122]. The interval arithmetic equivalent is also relatively straight forward to derive.

A more thorough overview of accurate summation and dot product techniques can be found in [116] and [46] in the floating point computation context, and [122] in the interval computation context.

Algorithm 13 s = CascadedDotProduct(x, y)

1: $A = \emptyset$
2: **for all** i = 1 to n **do**
3: p, e_p = EF_Product(x_i,y_i)
4: A = CasAccum(A, p)
5: A = CasAccum(A, e_p)
6: **end for**
7: **return** s = SumAccum(A)

3.3 Calculating Rigorous Interval Bounds on $F_m(T)$

There are several numerical approaches for evaluating the one and two-electron repulsion integrals. The majority of these involve recursively decomposing ERIs into linear combinations of simpler integrals known as *reduced incomplete gamma functions* or $F_m(T)$ [103, 115, 44] for short

$$F_m(T) = \int_0^1 t^{2m} e^{-Tt^2} dt \tag{3.5}$$

where $T \in \mathbb{R}$ is a non-negative value related to the distance between atomic centers and the value of the Gaussian exponents. $m \in \mathbb{Z}$ is dependent on the total angular momentum of the component functions in the two-electron integral. That is, supposing that L_a denotes the total angular momentum of χ_a, then the integral $(\chi_a\chi_b|\chi_c\chi_d)$ typically requires the computation of $F_m(T)$ values for $m = \{0, 1, 2, 3, .., L_a + L_b + L_c + L_d\}$ [38].

m	0	2	4	6	8	10	12	14	16	18
T_F	33	41	46	51	56	61	66	70	74	79

Table 3.1: Values of T_F as a function of even-values of m

The relation

$$F_{m+1}(T) = -\frac{d}{dT} F_m(T) \tag{3.6}$$

leads to the following recursive relation which can be used to compute $F_m(T)$ values [139]

$$F_m(T) = \frac{1}{2m+1}\{e^{-T} + 2TF_{m+1}(T)\}. \tag{3.7}$$

For practical reasons this expression can only be approximated numerically. The particular numerical approach which is applied depends on the value of T with respect to a threshold value denoted as T_F [139]. Table 3.1 summarises the value of T_F with respect to m. If $T > T_F$, then $F_m(T)$ is evaluated using the *asymptotic expression* [139],

$$\begin{aligned} F_m(T) &\approx \frac{1}{2}\sqrt{\frac{\pi}{T}}\frac{(2m-1)!!}{(2T)^m} \\ &= F_m^{Asymp}(T). \end{aligned} \tag{3.8}$$

On the other hand, if $T \leq T_F$, the recursive relation is unrolled to form an infinite series known as the *Shavitt series* [139]. However, due to computational considerations, $F_m(T)$ values within this range are usually approximated using either Taylor or Chebyshev interpolation.

To generate rigorous error bounds, it is necessary to bound the truncation errors introduced by the numerical approaches used to compute $F_m(T)$. We can achieve this by defining an interval extension of $F_m(T)$ that accounts for these errors.

We define the interval extension of $F_m(T)$ as

$$F_m(T^I) \supseteq \{F_m(T) : T \in T^I\}. \tag{3.9}$$

We assume that T^I is a non-degenerate interval that bounds T. If $F_m(T)$ can be defined as a rational function, then a natural interval extension will suffice. However, when numerical approximations are involved, the truncation errors that they introduce must also be accounted for in the way in which the interval extension is defined.

Without any loss of generality, suppose that $F_m(T^I)'$ is the natural interval extension of an approximation function of $F_m(T)$, and ϵ is the upper bound of the truncation error in T^I, then the interval extension of $F_m(T)$ can be written as

$$F_m(T^I) = [\underline{F_m(T^I)'} - \epsilon,\ \overline{F_m(T^I)'} + \epsilon]. \tag{3.10}$$

The interval produced using this function provide rigorous bounds on both the rounding and truncation errors resulting from the evaluation of $F_m(T)$.

The following section discusses how ϵ can be calculated for each of the various approaches used to evaluate $F_m(T)$.

3.3.1 The Shavitt Series

Shavitt's series is named after Shavitt [139] who first proposed it in 1963. Although initially an infinite series formed by unrolling Equation 3.7, it is usually truncated after a sufficient number of terms (n) have been evaluated

$$\begin{aligned} F_m(T) &= \frac{1}{2m+1}\{e^{-T} + 2TF_{m+1}(T)\} \\ &= \frac{1}{2m+1}\{\frac{2T}{2m+3}[\frac{2T}{2m+5}(..) + e^{-T}]\} \\ &= e^{-T}\sum_{i=0}^{\infty} \frac{(2T)^i}{(2m+1)(2m+3)...(2m+2i+1)} \\ &\approx e^{-T}\sum_{i=0}^{n} \frac{(2T)^i}{(2m+1)(2m+3)...(2m+2i+1)} \\ &= F_m^{Shav}(T). \end{aligned} \tag{3.11}$$

The truncation error of $F_m^{Shav}(T)$ can be expressed as

$$\begin{aligned} Err &= e^{-T} \sum_{i=n+1}^{\infty} \frac{(2T)^i}{(2m+1)(2m+3)...(2m+2i+1)} \\ &= e^{-T}\Big[\sum_{i=n+1}^{k} \frac{(2T)^i}{(2m+1)(2m+3)...(2m+2i+1)} + \\ &\qquad \sum_{i=k+1}^{\infty} \frac{(2T)^i}{(2m+1)(2m+3)...(2m+2i+1)}\Big] \\ &\leq e^{-T}\Big[\sum_{i=n+1}^{k} \frac{(2T)^i}{(2m+1)(2m+3)...(2m+2i+1)} + \\ &\qquad \sum_{j=0}^{\infty} \frac{(2T)^{k+1}}{(2m+1)(2m+3)...(2m+2k+3)}\Big(\frac{2T}{2m+2k+3}\Big)^j\Big] \\ &= e^{-T}\Big[\sum_{i=n+1}^{k} \frac{(2T)^i}{(2m+1)(2m+3)...(2m+2i+1)} + \\ &\qquad \frac{(2T)^{k+1}}{(2m+1)(2m+3)...(2m+2k+3)}\Big(\frac{2m+2k+3}{2m+2k+3-2T}\Big)\Big] \\ &= E_n^{Shav} \end{aligned} \tag{3.12}$$

where $k \in \mathbb{Z}^+$ is sufficiently large such that

$$\frac{2T}{2m+2k+3} < 1.0$$

hence yielding a series bounded by a geometric series with finite sum. Note that

the value of E_n^{Shav} is always positive, and therefore $F_m^{Shav}(T)$ is a lower bound of $F_m(T)$.

Supposing that $F_m^{Shav}(T^I)'$ is the natural interval extension of $F_m^{Shav}(T)$ in $T^I \in I(\mathbb{R})$ then the error bound of $F_m^{Shav}(T)$ for all $T \in T^I$ can be expressed as

$$F_m^{Shav}(T^I) = [\underline{F_m^{Shav}(T^I)'}, \overline{F_m^{Shav}(T^I)'} + E_n^{Shav}]. \tag{3.13}$$

Algorithm 14 shows the procedure for computing Shavitt's series. Observe that it is not too dissimilar to a dot product, but for the fact that each term is a multiplicative factor ($\frac{2T}{2m+1}$) of the previous term. It is therefore possible to implement Shavitt's series using modified variants of compensated summation and cascaded accumulated summation.

Algorithm 14 $s = F_m(T)^{Shav}$

1: $f = \frac{1}{2m+1}$
2: $m = m + 1$
3: $s = 0$
4: **repeat**
5: $\quad f = \otimes_\circ(f, \frac{2T}{2m+1})$
6: $\quad s = \oplus_\circ(s, f)$
7: $\quad m = m + 1$
8: **until** Convergence
9: **return** s

The compensated summation implementation of Shavitt's series is described in Algorithm 15 with respect to floating point arithmetic, and Algorithm 16 with respect to interval arithmetic. The idea is to compute correction terms for the multiply and add operations in each iteration (using error free transformation), i.e.

- $f = \otimes_\circ(f, g)$, and
- $s = \oplus_\circ(s, f)$

where $g = \frac{2T}{2m+1}$ - and then apply the corrections to the next iteration. For practical reasons, this approach does not correct the rounding errors from evaluating g, or f. A cascaded accumulated summation equivalent can be implemented based on the same idea.

Algorithm 15 $s = F_m(T)^{Shav} - Compensated$

1: $f = \dfrac{1}{2m+1}$
2: $m = m + 1$
3: $s = 0$
4: **repeat**
5: f, e_f = EF_Product(f, $\dfrac{2T}{2m+1}$)
6: $y = f + (e_f + e_s)$
7: s, e_s = EF_Add(s, y)
8: $m = m + 1$
9: **until** Convergence
10: **return** s

Algorithm 16 $S = F_m(T^I)^{Shav} - Compensated$

1: $F = \dfrac{1}{2m+1}$
2: $m = m + 1$
3: $e_{f_1}, e_{f_2} = 0$
4: $e_{s_1}, e_{s_2} = 0$
5: **repeat**
6: $G = \dfrac{2T^I}{2m+1}$
7: f_1, e_{f_1} = EF_Product($\underline{F}$, $\underline{G}$)
8: f_2, e_{f_2} = EF_Product($\overline{F}$, $\overline{G}$)
9: $F = F \times G$
10: $y_1 = \oplus_\Delta(-f_1, \oplus_\Delta(\text{-}e_{f_1}, e_{s_1},))$
11: $y_2 = \oplus_\Delta(f_2, \oplus_\Delta(e_{f_2}, e_{s_2},))$
12: s_1, e_{s_1} = EF_Add(s_1, y_1)
13: s_2, e_{s_2} = EF_Add(s_2, y_2)
14: $m = m + 1$
15: **until** Convergence
16: **return** $S = [-s_1, s_2]$

3.3.2 Polynomial Interpolation

A typical Hartree-Fock calculation requires millions of unique $F_m(T)$ values. It is therefore not efficient to use Shavitt's series, since hundreds of terms may need to be evaluated for each $F_m(T)$ value. A more efficient approach is to parametrize $F_m(T)$ using interpolated polynomial coefficients; using Shavitt's series only to evaluate the interpolation points. The coefficients of these polynomials can be pre-computed and stored in an interpolation table, which is accessed each time an $F_m(T)$ value is required.

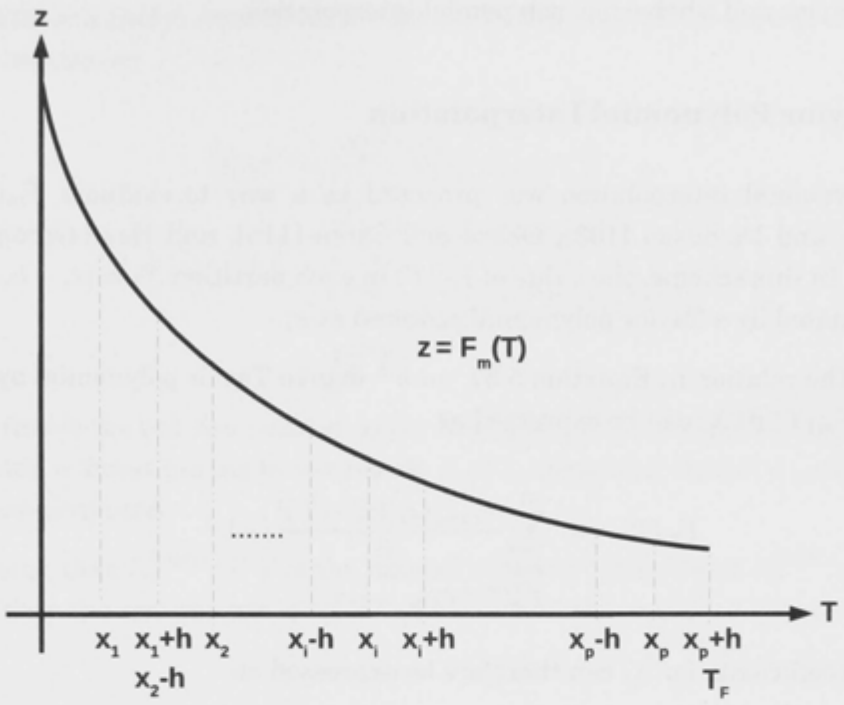


Figure 3.2: The uniform discretization of $z = F_m(T)$ for $T \in [0, T_f]$.

The basic procedure for generating an $F_m(T)$ interpolation table is described in Algorithm 17. In this procedure, the domain $[0, T_F]$ is discretized into p uniform partitions denoted as $X_i = [x_i - h, x_i + h], \ \forall i \in \{1, 2, ..p\}$ (See Figure 3.2). An n^{th}-degree polynomial interpolation of $F_m(T)$ is performed in each partition X_i, and the resulting coefficients $\{a_{ij}\}_{j=1}^{n}$ added to the interpolation table. For example, when $F_m(T)$ for $T \in X_i$ is required, $\{a_{ij}\}_{j=1}^{n}$ is accessed from the interpolation table in order to compute a polynomial approximation. As we shall see, the width of the discretization h, and by implication the size of the interpolation table, depends on i) the level of accuracy required, ii) the largest value of m required, and iii) the particular interpolation scheme used. Taylor [103] and Chebyshev [38] polynomial interpolation are the most popular schemes applied for this purpose.

Algorithm 17 Generate Interpolation Table

1: Discretize $[0, T_f]$ into small partitions $X_i = [x_i - h, x_i + h], \ \forall i \in \{1, 2, ..p\}$
2: **for** $i \in \{1, 2, ..p\}$ **do**
3: Interpolate $F_m(T)$ for $T \in X_i$ to obtain a set of polynomial coefficients $\{a_{ij}\}_{j=1}^{n}$ that describe $F_m(T)$ in X_i.
4: Append $\{a_{ij}\}_{j=1}^{n}$ to the interpolation table.
5: **end for**

The following sections outlines methods for bounding truncation errors intro-

duced by Taylor and Chebyshev polynomial interpolation.

3.3.3 Taylor Polynomial Interpolation

Taylor polynomial interpolation was proposed as a way to evaluate $F_m(T)$ by McMurchie and Davidson [103], Obara and Saika [115], and Head-Gordon and Pople [44]. In this scheme, the value of $F_m(T)$ in each partition $X_i = [x_i - h, x_i + h]$ is approximated by a Taylor polynomial centered at x_i.

Using the relation in Equation 5.37, an n^{th} degree Taylor polynomial approximation of $F_m(T)$ in X_i can be expressed as

$$\begin{aligned} F_m(T) &\approx \sum_{j=0}^{n} \frac{F_{m+j}(x_i)(x_i - T)^j}{j!}(-1)^j \\ &= F_m^{Taylor}(T). \end{aligned} \tag{3.14}$$

The set of coefficients for X_i can therefore be expressed as

$$a_{ij} = \frac{F_{m+j}(x_i)}{j!}(-1)^j, \ \forall j \in \{1, 2, ..n\}. \tag{3.15}$$

The $F_m(T)$ values at the points $T = x_i, \forall i \in \{1, 2, .., p\}$ are obtained by evaluating Shavitt's series. In order to reduce the number of operations, $F_m^{Taylor}(T)$ is usually evaluated using Horner's method [38]. For $T \in X_i$, $F_m^{Taylor}(T)$ can be written in Horner form as

$$F_m^{Taylor}(T) = a_{i0} + y(a_{i1} + y(a_{i2} + y(a_{i3} + y(a_{i4} + (...))))) \tag{3.16}$$

where $y = x_i - T$.

The rounding errors in evaluating each coefficient can be bounded by computing the natural interval extension of Equation 3.15 (which is a rational function).

The truncation error of $F_m^{Taylor}(T)$ is given by the *Lagrangian remainder* term. For an n^{th}-degree Taylor series approximation of $F_m(T)$, $T \in X_i$, this is written as

$$Err = \frac{F_m^{n+1}(\eta)}{(n+1)!}(T - x_i)^{n+1} \tag{3.17}$$

where $\eta \in \mathbb{R}$ is an unknown value defined in X_i. $F_m(T)$ is a monotonically decreasing function and $F_m(x_i - h) \geq F_m(\eta)$. Therefore the upper bound of the

(absolute) truncation error of an n^{th}-degree Taylor polynomial approximation in X_i can be written as

$$\begin{aligned} |Err| &= \left| \frac{F_m^{n+1}(\eta)}{(n+1)!}(T - x_i)^{n+1} \right| \\ &= \left| \frac{F_{m+n+1}(\eta)}{(n+1)!}(T - x_i)^{n+1} \right| \\ &\leq \left| \frac{F_{m+n+1}(x_i - h)}{(n+1)!}(T - x_i)^{n+1} \right| \\ &= E_n^{Taylor}. \end{aligned} \tag{3.18}$$

While this is the not the tightest upper bound of Err, it is useful in that it can be evaluated without having to recompute $F_m(T)$, supposing that $F_{m+n+1}(x_i - h)$ has been pre-computed.

Supposing that $F_m^{Taylor}(T^I)'$ is the natural interval extension of $F_m^{Taylor}(T)$ in $T^I \in I(\mathbb{R})^n$, then the error bound of $F_m^{Taylor}(T)$ for all $T \in T^I$, and $T^I \in X_i$, can be written as

$$F_m^{Taylor}(T^I) = [\underline{F_m^{Taylor}(T^I)' - E_n^{Taylor}}, \overline{F_m^{Taylor}(T^I)' + E_n^{Taylor}}]. \tag{3.19}$$

3.3.4 Chebyshev Polynomial Interpolation

Chebyshev interpolation defines the approximation of a function $f(x)$ for $x \in [-1, 1]$ as an expansion of *Chebyshev polynomials* $T_i(x)$ as follows

$$f(x) \approx \sum_{i=0}^{n} \beta_i T_i(x) \tag{3.20}$$

where

$$\begin{aligned} T_0(x) &= 1 \\ T_1(x) &= x \\ T_2(x) &= 2x^2 - 1 \\ T_3(x) &= 4x^3 - 3x \\ T_4(x) &= 8x^4 - 8x^2 + 1 \\ &\ldots \\ T_{n+1}(x) &= 2xT_n(x) - T_{n-1}(x) \qquad n \geq 1 \\ &= \cos\left((n+1)\arccos(x)\right). \end{aligned} \tag{3.21}$$

In the domain $[-1, 1]$, Chebyshev polynomials are orthogonal under integration, have a range of $[-1, 1]$, and have n distinct roots known as *Chebyshev zeroes*. These

properties are used to derive expressions for the Chebyshev polynomial coefficients, which are written as follows

$$\beta_i = \begin{cases} \dfrac{1}{n+1}\displaystyle\sum_{j=0}^{n} f\Big(\cos\Big(\frac{\pi(k-0.5)}{n}\Big)\Big), & \text{if } i = 0 \\ \dfrac{2}{n+1}\displaystyle\sum_{j=0}^{n} f\Big(\cos\Big(\frac{\pi(k-0.5)}{n}\Big)\Big) \times \Big[\frac{\pi i(k-0.5)}{n}\Big] & \text{if } i > 0. \end{cases} \tag{3.22}$$

The n^{th}-degree Chebyshev polynomial expansion of $F_m(T)$ in X_i is written as

$$\begin{aligned} F_m(T) &\approx \sum_{j=0}^{n} a_{ij} T_j(T) \\ &= F_m^{Cheby}(T). \end{aligned} \tag{3.23}$$

A modified Chebyshev interpolation scheme for approximating $F_m(T), T \in X_i$ was proposed by Gill [38]. In this scheme the polynomial coefficients are defined as

$$a_{ij} = \begin{cases} \displaystyle\sum_{k=0}^{\lfloor N/2 \rfloor} \Big(\frac{h}{2}\Big)^{2k} \frac{F_m^{(2k)}(x_i)}{k!k!} & \text{if } j = 0 \\ 2\Big(\frac{h}{2}\Big)^{j} \displaystyle\sum_{k=0}^{\lfloor (N-j)/2 \rfloor} \Big(\frac{h}{2}\Big)^{j+2k} \frac{F_m^{(j+2k)}(x_i)}{k!(j+k)!} & \text{if } j > 0 \end{cases} \tag{3.24}$$

The advantage of this scheme is that only a single interpolation point needs to be evaluated explicitly per partition, the other points can be derived from Equation 3.7.

For an n^{th}-degree interpolation, the above formulation is derived from the following steps

1. Expand $F_m(T)$ in terms of an N^{th}-degree Taylor polynomial, where $N > n$.
2. Reformulate the Taylor polynomial as an N^{th}-degree Chebyshev polynomial defined by coefficients $\{a_{ij}\}_{j=0}^{N}$.
3. Truncate the N^{th}-degree Chebyshev polynomial at the n^{th} degree, yielding a set of polynomial coefficients $\{a_{ij}\}_{j=0}^{n}$.

Since $T_n(x) \in [-1, 1]$, the upper bound of the truncation error can be written as the sum of the truncated coefficients and the truncation error of the N^{th}-degree

Taylor polynomial. For an n^{th}-degree Chebyshev approximation in X_i, the absolute truncation error can be written as

$$\begin{aligned} |Err| &\leq |a_{i,n+1}| + |a_{i,n+2}| + ... + |a_{i,N}| + E_N^{Taylor} \\ &= E_n^{Cheby} \end{aligned} \tag{3.25}$$

where E_N^{Taylor} is defined in Equation 3.18. Note also that the set of truncated coefficients can be pre-computed.

Supposing that $F_m^{Cheby}(T^I)'$ is the natural interval extension of $F_m^{Cheby}(T)$ in $T \in I(\mathbb{R})^n$, then the error bounds of $F_m^{Cheby}(T)$ for all $T \in T^I$, and $T^I \in X_i$, can be written as

$$F_m^{Cheby}(T^I) = [\underline{F_m^{Cheby}(T^I)'} - E_n^{Cheby}, \overline{F_m^{Cheby}(T^I)'} + E_n^{Cheby}] \tag{3.26}$$

$F_m^{Cheby}(T)$ can be evaluated using either *Clenshaw's Recurrence* for greater accuracy, or using Horner's method for greater efficiency. An efficient scheme for evaluating third degree Chebyshev polynomials using Horner's method is given in [38] as

$$F_m^{Cheby}(T) = f_0 + \frac{T}{2h}(f_1 + \frac{T}{2h}(f_2 + \frac{T}{2h})) \tag{3.27}$$

where in X_i, f_0, f_1, f_2, and f_3 are defined as

$$\begin{aligned} f_0 &= (a_{i0} - a_{i2}) + (a_{i1} - 3a_{i3})(\frac{-x_i}{h}) + 2a_{i2}(\frac{-x_i}{h})^2 + 4a_{i3}(\frac{-x_i}{h})^3 \\ f_1 &= (2a_{i1} - 6a_{i3}) + 8a_{i2}(\frac{-x_i}{h}) + 24a_{i3}(\frac{-x_i}{h})^2 \\ f_2 &= 8a_{i2} + 48a_{i3}(\frac{-x_i}{h}) \\ f_3 &= 32a_{i3} \end{aligned} \tag{3.28}$$

3.3.5 The Asymptotic Expression

The asymptotic expression for evaluating $F_m(T)$ when $T > T_f$ is given in Equation 3.8. It is derived by splitting the integral defining $F_m(T)$ into two integrals, one infinite and the other finite

$$\begin{aligned} F_m(T) &= \int_0^\infty t^{2m} e^{-Tt^2} dt - \int_1^\infty t^{2m} e^{-Tt^2} dt \\ &= \frac{1}{2}\sqrt{\frac{\pi}{T}} \frac{(2m-1)!!}{(2T)^m} - \int_1^\infty t^{2m} e^{-Tt^2} dt \\ &= F_m^{Asymp}(T) - Err \end{aligned} \tag{3.29}$$

The integral on the left is equivalent to the *complete gamma function*, which can be evaluated analytically to obtain the asymptotic expression. Whereas the integral on the right is the truncation error term, which can be re-written as [139]

$$\begin{aligned} Err \;\leq\; & e^{-T} \sum_{i=0}^{m} \Bigg[\frac{(2m-1)(2m-3)..(2(m-i)+1)}{(2T)^i} + \\ & \frac{e^{-T}}{2T} \frac{(2m-1)(2m-3)..(2(m-i)+1)}{(2T)^i} \Bigg] \\ =\; & E^{Asymp} \end{aligned} \tag{3.30}$$

Supposing that $F_m^{Asymp}(T^I)'$ is the natural interval extension of $F_m^{Asymp}(T)$ in $T^I \in I(\mathbb{R})^n$, then the error bounds of $F_m^{Asymp}(T)$ for all $T \in T^I$, and $T^I > T_F$, can be written as

$$F_m^{Asymp}(T^I) = [\underline{F_m^{Asymp}(T^I)'} - E^{Asymp}, \overline{F_m^{Asymp}(T^I)'}] \tag{3.31}$$

3.4 Calculating Rigorous Interval Bounds on the Total Energy

The operations required to calculate the one and two-electron integrals and then the total energy can be expressed as rational functions with frequent evaluations of $F_m(T)$. It is therefore possible (by Definition 2.21), to rigorously bound numerical errors in these quantities simply by evaluating their natural interval extensions, and relying on the interval extensions of $F_m(T)$ defined in the previous section.

The procedure to evaluate the Fock matrix (Equation 2.15) and the electronic energy (Equation 2.18) are both numerically expensive steps similar to summations and dot products. *Interval Hartree-Fock* provides an option to compute these quantities using compensated summation and cascaded accumulated summation, which can be invoked when tighter interval bounds are required.

3.5 Diagonalization of the Interval Fock Matrix

In the Self-Consistent Field method, the Fock matrix is diagonalized to produce a molecular orbital coefficient matrix, which is then used to calculate a new Fock matrix. This process is repeated until the molecular orbital coefficient matrix converges towards a fixed point corresponding either to the ground state, or another stable energy state. Fock matrix diagonalization also yields additional information about the molecular system being considered. For example, the eigenvalues of the Fock matrix correspond to the molecular orbital energies, while the eigenvectors of the Fock matrix correspond to the molecular orbital coefficients [148]. The latter can also be used to derive electronic properties and for population analyses [148].

The successful convergence of the SCF procedure, and the accuracy of each quantity derived from the Fock matrix are inevitably affected by numerical errors propagated by Fock matrix diagonalization. This is caused by numerical errors in the earlier computational steps, such as in the evaluation of the one and two-electron integrals which are used to evaluate the Fock matrix. An error contaminated Fock matrix of this kind can be represented as a symmetric *interval Fock matrix*, where the numerical errors in each element is bounded by an interval.

The following section introduces approaches for diagonalizing the interval Fock matrix: first by outlining methods for computing bounds on each of its eigenvalues, and second by describing methods for finding the bounds on each of its eigenvectors. This allows us to bound the numerical errors propagated by Fock matrix diagonalization, and therefore compute error bounds on quantities such as the molecular orbital energies, the molecular orbital coefficients, and derived electronic properties.

3.5.1 Notations

For the following sections, we generally adopt the interval matrix and eigenvalue notations used in the work by Hladik, Daney, and Tsigaridas [47]. The definitions of interval matrices and interval vectors were previously discussed in Section 2.4.

Let $\mathbf{A} \in I(\mathbb{R}^{n\times n})$ be a square interval matrix. The notation $\mathbf{A} = [\underline{A}, \overline{A}]$, $\underline{A}, \overline{A} \in \mathbb{R}^{n\times n}$ is defined such that

$$\mathbf{A} = \{A \in \mathbb{R}^{n\times n} : \underline{A} \leq A \leq \overline{A}\}$$

where $\underline{A}$ and $\overline{A}$ can be interpreted as real matrices representing the lower and

upper endpoints of **A**, respectively. This notation is useful for explaining the following quantities.

The *center matrix* of **A** is denoted as $A_c \in \mathbb{R}^{n\times n}$ where

$$A_c = \frac{1}{2}(\overline{A} + \underline{A})$$

The *radius matrix* of **A** is denoted as $A_\triangle \in \mathbb{R}^{n\times n}$ where

$$A_\triangle = \frac{1}{2}(\overline{A} - \underline{A})$$

Notice that in this section interval matrices and vectors are denoted in bold font, unless stated otherwise.

Let **A** be an interval matrix where $\underline{A}$ and $\overline{A}$ are symmetric. A *symmetric interval matrix* $\mathbf{A}^s \in I(\mathbb{R}^{n\times n})$ is defined such that

$$\mathbf{A}^s = \{A \in \mathbf{A} : A = A^T\}$$

In practice given that intervals do not retain information about the dependence between matrix elements, there are subsets of $\mathbf{A}^s$ that are not symmetric.

The interval Fock matrix is a symmetric interval matrix.

A real matrix $A \in \mathbb{R}^{n\times n}$ may have up to n real eigenvalues ordered as follows

$$\lambda_1 \geq \lambda_2 \geq ... \geq \lambda_n$$

where $\lambda_i \in \mathbb{R}^n$ is referred to as the i^{th} eigenvalue of **A**. Note that when eigenvalues of different matrices exist in the same context we use the notation $\lambda_i(A)$ to denote the i^{th} eigenvalue of A, in order to avoid any confusion.

The eigenvector x_i corresponding to λ_i is defined as

$$\begin{aligned} Ax_i &= \lambda_i x_i \\ (A - I\lambda_i)x_i &= 0 \end{aligned}$$

The set of all real eigenvalues of an interval matrix $\mathbf{A} \in I(\mathbb{R}^n)$ is defined as

$$\Lambda = \{\lambda \in \mathbb{R} : Ax = \lambda x,\ x \neq 0, A \in \mathbf{A}\}$$

where Λ is a compact set, composed of up to n compact real intervals, each corresponding to an *interval eigenvalue* [47].

3.5.2 Finding the Eigenvalues of a Symmetric Interval Matrix

The problem of finding the exact interval hull containing Λ is generally NP-Hard [47]. However, an interval λ^0 that provides an outer approximation of Λ, such that $\Lambda \subseteq \lambda^0 = [\underline{\lambda^0}, \overline{\lambda^0}]$, can be computed feasibly using the following relation given by Rohn [123],

$$\begin{aligned} \underline{\lambda^0} &= \lambda_{min}(A_c) - \rho(A_\triangle) \\ \overline{\lambda^0} &= \lambda_{max}(A_c) + \rho(A_\triangle) \end{aligned} \tag{3.32}$$

where $\lambda_{min}(A_c)$ and $\lambda_{max}(A_c)$ are the smallest and largest eigenvalues of A_c; and $\rho(A_\triangle)$ is the spectral radius of $A_\triangle$. Alternatives to this scheme include methods based on Gerchgorin's theorem [36], and Brauer's Cassini ovals [14].

To obtain error bounds on individual molecular orbital energy values, we must compute error bounds on individual eigenvalues of the Fock matrix. The i^{th} *interval eigenvalue* of $\mathbf{A}$, is defined as an interval containing the following set

$$\lambda_i(\mathbf{A}) = \{\lambda_i(A) : A \in \mathbf{A}\}$$

Note that it is possible for interval eigenvalues to overlap with one another.

While there are generally no feasible way to calculate the exact range of an interval eigenvalue, it is possible to calculate an outer approximation which contains it.

For symmetric interval matrices the following formula given by Rohn [123] (proven in [47]) provides the basic bounds for $\lambda_i(\mathbf{A})$

$$\lambda_i(\mathbf{A}) \subseteq [\lambda_i(A_c) - \rho(A_\triangle)), \lambda_i(A_c) + \rho(A_\triangle)] \tag{3.33}$$

while conceptually simple, the bounds produced by this expression are usually sharp [47]. The drawback is that the absolute width of each interval eigenvalue is identical (equal to the spectral radius). Although more sophisticated approaches exist such as those based on *Cauchy's interlacing property* [47],

Theorem 3.1. ***(Cauchy's interlacing property)*** *Let $A \in \mathbb{R}^n$ be a symmetric matrix and let A_i be a matrix obtained from A by removing the i^{th} row and column of A, then*

$$\lambda_1(A) \geq \lambda_1(A_i) \geq \lambda_2(A) \geq \lambda_2(A_i) \geq ... \geq \lambda_{n-1}(A_i) \geq \lambda_n(A) \tag{3.34}$$

From [47].

they are only useful when some of the interval eigenvalues overlap, or when there is only a narrow gap between them [47]. This is generally not the case for Fock matrices composed of relatively narrow interval elements.

3.5.3 Finding the Eigenvectors of a Symmetric Interval Matrix

This section considers the problem of finding the *interval eigenvectors* of the Fock matrix, which can be used to produce error bounds on molecular orbital coefficients. The interval eigenvector $\mathbf{x_i} \in I(\mathbb{R}^n)$ that corresponds to the i^{th} interval eigenvalue of $\mathbf{A} \in I(\mathbb{R}^{n\times n})$ is defined as

$$\mathbf{x_i} = \{x \in \mathbb{R} : (A - I\lambda_i(A))x = 0,\ \forall A \in \mathbf{A}\} \tag{3.35}$$

Like the interval eigenvalues discussed previously, it is only feasible to compute an outer approximation of $\mathbf{x_i}$. This problem is equivalent to finding the interval vector $\mathbf{z} \in I(\mathbb{R})^n$ that encloses the solution of an *interval system of linear equations* [42] $\mathbf{A'x} = 0$

$$\mathbf{x} = \{x \in \mathbb{R} : A'x = 0,\ A' \in \mathbf{A'}\} \tag{3.36}$$

where $\mathbf{A'} = (\mathbf{A} - I\lambda_i(\mathbf{A}))$, $\mathbf{A'} \in I(\mathbb{R}^n)$, $\lambda_i(\mathbf{A}) \in I(\mathbb{R}^n)$, and $\mathbf{x} \in I(\mathbb{R}^n)$.

Adding a constraint that excludes the degenerate solution, $\mathbf{x} = 0$, leads to an overdetermined system of the form $\mathbf{Bx} = \mathbf{b}$, $\mathbf{B} \in I(\mathbb{R}^{(n+1)\times n})$, $\mathbf{b} \in I(\mathbb{R}^{n+1})$ where

$$\mathbf{B} = \begin{bmatrix} \mathbf{A'_{11}} & \mathbf{A'_{12}} & \dots & \mathbf{A'_{1n}} \\ \mathbf{A'_{21}} & \mathbf{A'_{22}} & \dots & \mathbf{A'_{2n}} \\ \vdots & \vdots & \vdots & \ddots \\ \mathbf{A'_{n1}} & \mathbf{A'_{n2}} & \dots & \mathbf{A'_{nn}} \\ 1 & 1 & \dots & 1 \end{bmatrix} \quad \text{and} \quad \mathbf{b} = \begin{bmatrix} 0 \\ 0 \\ \vdots \\ 0 \\ 1 \end{bmatrix} \tag{3.37}$$

The overdetermined system can be recast as an equivalent square system of the form

$$\begin{bmatrix} \mathbf{B} & I \\ 0 & \mathbf{B^T} \end{bmatrix} \begin{bmatrix} \mathbf{x} \\ \mathbf{y} \end{bmatrix} = \begin{bmatrix} \mathbf{b} \\ 0 \end{bmatrix} \tag{3.38}$$

The application of approaches such as Gaussian elimination to this problem leads to large overestimations of $\mathbf{x}$ [42, 66]. However, specific approaches for solving interval systems of linear equations have been developed over the years.

Extensive reviews on this topic found in [113, 42, 66]. The following sections briefly discuss two well known methods: the ubiquitous *Krawczyk's Iteration* developed by Krawczyk [74] and Moore [109] and the more recent *Shary's Method* outlined by Shary [138]. We should also mention the *Interval Gauss-Seidel Method* which is another often used approach, explained in [65, 43, 66] and elsewhere.

Krawczyk's Iteration

Consider the matrix problem of the form $Ax = b$, where $A \in \mathbb{R}^{n\times n}$, $b \in \mathbb{R}^n$, and $x \in \mathbb{R}^n$. If A has an inverse A^{-1}, then $Ax = b$ can be re-written as

$$\begin{aligned} x &= A^{-1}b \\ &= Cb + (I - CA)A^{-1}b \\ &= Cb + (I - CA)x \end{aligned} \tag{3.39}$$

where $C \in \mathbb{R}^{n\times n}$ is an arbitrary *preconditioning matrix*.

Therefore, the equivalent interval system of linear equations, $\mathbf{Ax} = \mathbf{b}$, can be expressed by the fixed point equation of the form

$$\mathbf{x} \subseteq C\mathbf{b} + (I - C\mathbf{A})\mathbf{x} \tag{3.40}$$

where $\mathbf{A} \in I(\mathbb{R}^{n\times n})$, $\mathbf{b} \in I(\mathbb{R}^n)$, $C \in \mathbb{R}^{n\times n}$, and $\mathbf{x} \in I(\mathbb{R}^n)$.

Letting $\mathbf{z_k} \in \mathbb{R}^n$ be an interval vector that encloses x, such that $\mathbf{x} \subseteq \mathbf{z_k}$, then Equation 3.40 can be expressed as the following fixed point iteration

$$\mathbf{z_{k+1}} = \mathbf{Kz_k} \bigcap \mathbf{z_k} \tag{3.41}$$

K is the Krawczyk operator defined as

$$\mathbf{K} = C\mathbf{b} + (I - C\mathbf{A}) \tag{3.42}$$

Brouwer's fixed point theorem [42] states that, i) if $\mathbf{Kz_k} \bigcap \mathbf{z_k} = \emptyset$ then there exists no solution in $\mathbf{z_k}$, and ii) if $\mathbf{Kz_k} \subset \mathbf{z_k}$ then there exists a unique solution in $\mathbf{z_k}$. Furthermore, assuming that $\mathbf{z_k}$ contains x, then i) $\mathbf{Kz_k}$ must also contain x given the relation in Equation 3.40, and ii) a subset of $\mathbf{Kz_k}$ that does not contain $\mathbf{z_k}$ cannot possibly contain x. These relations, together, imply that the enclosure of x can be tightened by computing the intersection of the sets spanned by $\mathbf{Kz_k}$ and $\mathbf{z_k}$.

The Krawczyk's iteration scheme uses Equation 3.41 to iteratively improve the enclosure of x; starting with an initial guess of $z_0 \supseteq x$ and stopping after a termination criteria is satisfied. This ideally gives rise to a convergent sequence of enclosures: $z_0 \supseteq z_1 \supseteq z_2 \supseteq z_3 \supseteq \ldots\ldots \supseteq x$.

More effective, modified, versions of Krawczyk's iteration have since been developed, including those introduced by Hansen [41], Rump [126], and Jansson [60].

Shary's Method

Shary [138] employed an *algebraic approach* to bounding the solution of a system of interval linear equations. This involves formulating the original problem as an interval fixed point iteration of the form

$$z_{k+1} = Mz_k + G^{-1}b \tag{3.43}$$

and then mapping it into an equivalent algebraic system of the form $\Phi(z) = 0$, which is then solved using a variation of Newton's method referred to as the *Subdifferential Newton's method*. By avoiding interval computation, Shary's method is able to generate very tight enclosures compared to other contemporary approaches [138].

The details of the interval to algebra mapping, and Shary's method in general, are provided in [138, 110].

3.6 Implementation Details

The methods discussed in this chapter are implemented using a combination of C++ and SPARC assembly code, totalling over 20,000 lines. Interval operations are performed using the SunStudio 11 C++ Interval Arithmetic Library [144]. Large scale matrix calculations are performed using optimized linear algebra routines from the Sun Studio 11 Performance Library [145]. The entire implementation is collectively referred to as the *Interval Hartree-Fock* program.

Provided an input specifying a closed-shell molecular system, and the current electronic state, this *Interval Hartree-Fock* is able to compute rigorous error bounds on quantities including i) the total energy, ii) the atomic orbital energies, iii) the molecular orbitals, and iv) derived electronic properties.

The different components of this program are described below. Unless stated otherwise, it should be assumed that all operations mentioned are performed using interval arithmetic.

- **Parser** The program accepts molecular systems and basis sets specified in Gaussian03 input format. The program also accepts a separate configuration file, as well as parameters through command line arguments.

- $F_m(T)$ **Reduced Incomplete Gamma Function Evaluation** The program implements several different approaches for evaluating $F_m(T)$, based on the interval techniques developed in Section 3.3 to guarantee containment of numerical errors.

  Both the Taylor and Chebyshev polynomial interpolation schemes are implemented, based on the algorithms described in [38]. Three different versions of Shavitt series evaluation are implemented, one based on the conventional approach specified in Algorithm 14, and the other two based respectively on compensated summation and cascaded accumulated summation.

- **Electron Repulsion Integral Evaluation** The program implements multiple paths of the McMurchie-Davidson (MD) PRISM algorithm [103, 40], as well as the Head-Gordon-Pople (HGP) PRISM algorithm [44, 40]. Interval pre-screening based on Schwarz's upper bound [148] is also implemented.

- **Total Energy Evaluation** The Fock matrix and the total energy are evaluated based on Equation 2.15 and Equation 2.18, respectively. We also implemented code that allows these quantities to be computed using compensated summation and cascaded accumulated summation techniques.

- **Fock Matrix Diagonalization** Bounds on the eigenvalues of the interval Fock matrix are obtained by firstly performing Rohn's basic bounding approach (Equation 3.33), and then refining the bounds by applying the Cauchy interlacing-based approaches described by Hladik in [47]. After performing the transformations outlined in Section 3.5.3, the eigenvectors of the Fock matrix were obtained using Shary's sub-differential Newton method [138], using code made available by Shary [2].

  It should be noted that the Fock matrix diagonalization component was implemented using a mixture of floating point and interval code that may produce results affected by round-off. The task of implementing the entire diagonalization code using intervals is left to future work.

[2] http://old.ict.nsc.ru/rus/curvitae/shary/Codes (accessed 5 June 2011)

To complement this work, we also implemented a standard SCF solver capable of performing general closed-shell Hartree-Fock calculations in IEEE754 double precision arithmetic. SCF convergence is accelerated using Direct Inversion of Iterative Subspaces (DIIS) [119]. This program is provides the ground state molecular orbitals used as input for *Interval Hartree-Fock*.

The source code for *Interval Hartree-Fock* is available on the author's web site [56].

3.7 Conclusion

This chapter introduces methods for placing rigorous error bounds on Hartree-Fock quantities including the total energy, the molecular orbital energies, the molecular orbital coefficients, and derived molecular properties. These methods were then used to develop the *interval Hartree-Fock* program.

The chapter provides a brief background to accurate summation and dot product methods using compensated summation and cascaded accumulated summation. Although these methods are not necessary for generating error bounds, they are able to provide tighter error bounds by reducing the accumulation of rounding errors.

The chapter then focuses on the core issue of placing error bounds on specific Hartree-Fock quantities. This involved identifying the various computational steps where errors are introduced, and, for each, defining an appropriate interval extension to bound the range of perturbations due to numerical errors.

We developed methods for bounding numerical errors associated with computing $F_m(T)$, a core quantity used in the construction of one and two-electron integrals. This required defining interval extensions for i) Shavitt's series evaluation, ii) Taylor polynomial interpolation, iii) Chebyshev polynomial interpolation, and iv) the evaluation of the asymptotic expression. This then allowed rigorous error bounds on the Hartree-Fock total energy to then be evaluated.

Finally, we address the problem of diagonalizing Fock matrices with interval elements containing errors propagated from earlier computational steps. This is necessary in order to place error bounds on quantities such as the molecular orbital energies, the molecular orbital coefficients, and derived electronic properties. This was reduced into the problems of i) finding the bounds on all the eigenvalues of an interval matrix, and ii) finding the bounds on the solution of an interval system of linear equations.

In conclusion, this chapter contributes methods for producing verified bounds on the results of a Hartree-Fock calculation. These bounds are guaranteed to

contain the result that would have been produced had exact arithmetic and analytical approaches been possible. However, these bounds must also contain the possible range of perturbations due to numerical errors. This was primarily achieved through the application of interval analysis techniques.

In the next chapter, *Interval Hartree-Fock* will be applied as an error analysis tool to study the impact various input and design related factors on numerical errors in Hartree-Fock computation.

Chapter

4

An Analysis of Numerical Errors in Hartree-Fock Computation

Contents

4.1 Introduction

The immense computational effort required to perform electronic structure calculations has resulted in the adoption of methods of computation that tend to emphasise efficiency, even when there are concerns over its repercussions to the quality of the results produced. This is partly due to the belief that only an insignificant amount of numerical accuracy needs to be sacrificed in exchange for a vast improvement in the time to solution. However, in practice it is rarely possible to quantify the impact that such a trade-off may have until a serious problem occurs.

The *Interval Hartree-Fock* program developed in the previous chapter was designed to mitigate the above issue by providing rigorous bounds on numerical errors in Hartree-Fock computation.

In this chapter *Interval Hartree-Fock* will be applied as an error analysis tool to study the growth of numerical errors due to input related factors such as increasing basis set and system sizes. It will also be used to explore the impact of algorithmic design related factors on the accuracy of Hartree-Fock calculations, considering factors such as: i) the choice of $F_m(T)$ evaluation scheme, ii) the application of compensated summation and cascaded accumulated summation, iii) variations in the integral pre-screening threshold, iv) variations in the arithmetic precision in which integrals are evaluated, and v) the relaxation of error tolerances of interpolation tables.

Some of the issues considered above are not only pertinent to the usage of conventional Hartree-Fock programs, but also have special significance in informing the way in which Hartree-Fock code is implemented for novel computing architectures such as graphics processing units (GPU) and the STI Cell Broadband Engine (CellBE); for exascale and green computing environments; and for the exploitation of the emerging generation of massively multi-core processors.

This chapter is structured into the following sections. Section 4.2 characterizes the role of interval analysis as an error analysis tool. Section 4.3 outlines the

experimental platform used in this chapter. Section 4.4 investigates numerical errors in evaluating $F_m(T)$. Section 4.5 studies numerical errors in evaluating the Hartree-Fock total energy, considering the impact of increasing basis set and system sizes, as well as the choice of $F_m(T)$ evaluation scheme, and the application of accurate summation techniques. Section 4.6 studies the effect of integral screening on numerical errors in the total energy. Section 4.7 briefly reviews issues related to implementing Hartree-Fock code on specialized processing hardware such as graphics processing units (GPU). Section 4.8 investigates numerical errors in Hartree-Fock calculations performed on a GPU, considering i) the scenario where integrals are evaluated using a mixture of single and double precision floating point operations, and ii) the effect of varying the granularity of the $F_m(T)$ interpolation table. Section 4.9 examines numerical errors in the Fock matrix diagonalization step. Section 4.10 discusses related work. Finally, Section 4.11 concludes the chapter.

4.2 Interval Analysis as an Error Analysis Tool

Interval numbers can be used as a representation for the uncertainty associated with an individual quantity. In this context, the principles of interval arithmetic and interval functions, together, enable mathematical operations between uncertain quantities to take place in such a way that the cumulation of all uncertainties is contained fully within the bounds of the final result. The width of an interval can, therefore, be used to provide a *worst-case estimate* of the errors associated with a given computed quantity.

Suppose then that an uncertain quantity $z \in \mathbb{R}$ lies within an interval denoted as $Z \in I(\mathbb{R})$, then the *interval relative error* of z is expressed as the width of Z divided by its midpoint [105, 58, 122, 121] (see Definition 2.11)

$$Err = \frac{w(Z)}{m(Z)}. \tag{4.1}$$

This expression is independent of how z or Z is derived, the only assumption being that $z \in Z$. Suppose that z is defined by a real function $f(x)$, then Z can be derived from evaluating the interval extension of $f(x)$ denoted as $F(X)$. It is assumed that $F(X)$ guarantees containment with respect to the range of $f(x)$ in X, as well as any numerical error associated with computing this range.

The final interval, such as $F(X)$, will almost always overestimate the actual errors due to the fact that interval arithmetic must account for the worst possible interaction between interval quantities involved in the calculation. The equivalent

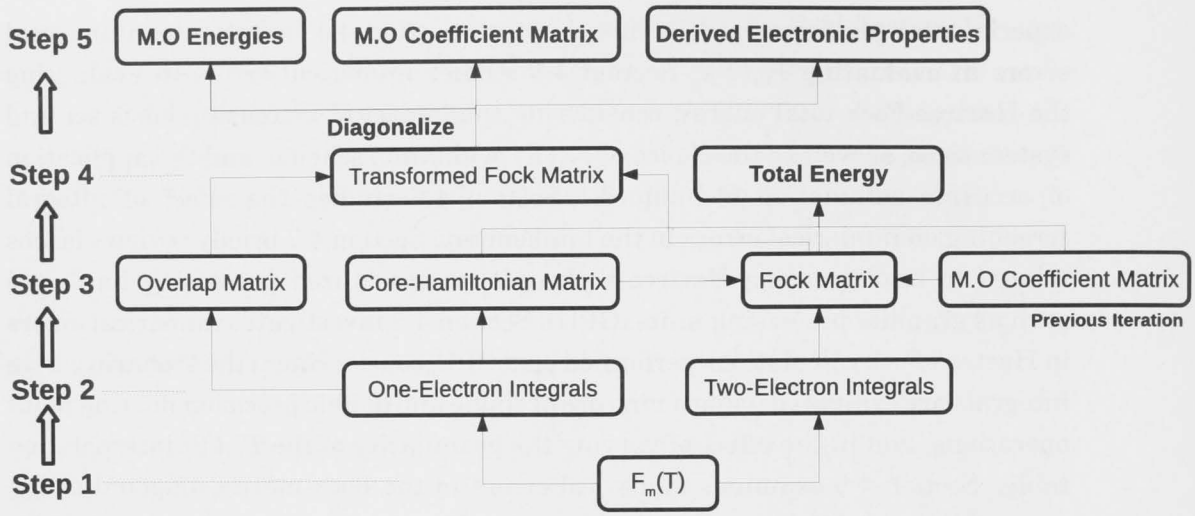


Figure 4.1: The computational steps involved in evaluating the Hartree-Fock i) total energy, ii) molecular orbital energies, iii) molecular orbital coefficients, and iv) derived electronic properties, given an initial set of molecular orbital coefficients.

problem of finding the exact range over a given domain of uncertainty is generally NP-Hard. This implies that interval arithmetic can only provide a *worst-case error analysis* of the calculation in question. Therefore, an interval can only rigorously state what set of factors can or cannot satisfy a given threshold of accuracy. For example, it is permissible to state that

The error of z is guaranteed to never exceed Err

In computational chemistry, this relation can be used to verify whether or not a particular result is within chemical accuracy. In the broader context, it is useful for demonstrating that certain parameters or constraints within a mathematical model are within specified tolerances. This type of analysis is sometimes referred to as *tolerance analysis*. All of this is not to say that other useful inferences cannot be derived from interval error bounds, but they must always be expressed with the *worst-case* caveat in mind.

Another issue that may also cause interval error overestimation is the dependency problem discussed in Section 2.4. However, dependency effects are likely to be small in this particular work, since we mainly deal with narrow intervals containing numerical errors.

4.2.1 Interval Analysis for Error Propagation

Figure 4.1 summarizes the computational steps required to evaluate quantities such as the *total energy*, the *orbital energies*, the *molecular orbital coefficients*, and *derived electronic properties*. The arrows represent the sequence in which the intermediate quantities are constructed, and eventually leads to the required quantities.

The first step involves the computation of the reduced incomplete gamma functions denoted as $F_m(T)$ (Section 3.3), which are then used i) to evaluate the *one-electron integrals* that contribute to the *core-Hamiltonian* (Equation 2.16) and the *overlap* matrices, and ii) evaluate the *two-electron repulsion integrals* (ERI) (Section 2.2.6). The one and two-electron integrals are used to construct the *Fock matrix* (Equation 2.15), which in combination with the *molecular orbital coefficients*, can then be used to evaluate the total energy (Equation 2.18 and Equation 2.3). The M.O energies, the M.O coefficients (for the next iteration), and derived molecular properties can be obtained by *transforming* (Equation 2.19) and then *diagonalizing* (Section 3.5) the Fock matrix.

Given the relationship between the different intermediate quantities, it is reasonable to expect that errors introduced in one will propagate on to another, and each eventually have an impact on the accuracy of the final result. Thus, Figure 4.1 can also be seen as an illustration of how numerical errors may propagate across a typical Hartree-Fock calculation.

Interval analysis provides the necessary mathematical tools to represent and propagate errors within a complicated model such as Hartree-Fock. The containment property of interval functions allows the errors introduced at any stage of the computation to propagate fully towards the final result. Conversely, the source of error can be attributed to a particular stage of computation by examining the error bounds produced at each stage through a process of elimination. These properties are common with uncertainty propagation techniques used in *uncertainty analysis* and *sensitivity analysis*.

4.2.1.1 Comparison to Uncertainty and Sensitivity Analysis

The purpose of the following section is to compare the interval analysis approach to error analysis with conventional uncertainty and sensitivity analysis techniques. We will base the discussion within the context of Hartree-Fock computation.

For clarity, it is best to discuss uncertainty and sensitivity analysis in terms of a generic multi-variate model of the form

$$y = f(x_1, x_2, .., x_N) \tag{4.2}$$

where $\{x_i\}_{i=1}^{N}$ are the set of uncertain input variables, and y is an uncertain output. In some models the uncertainties in each variable can be described in terms of a statistical distribution, e.g. $x_i = N(\langle x_i \rangle, \delta_i)$ and $y = N(\langle y \rangle, \delta)$, where for example $N(\langle x_i \rangle, \delta_i)$ denotes a normal distribution with a mean of $\langle x_i \rangle$ and variance of δ_i.

Uncertainty analysis is the study of variations in model output as a result of variations in the model inputs. Sensitivity analysis is the study of how the variation in the model output can be apportioned to the variations of different input variables - or in other words, it assesses the *sensitivity* of the model output to variations in different input variables [133].

Sensitivity analysis approaches can be characterized as being either *local* or *global* in nature [134, 133]. A local measure of the sensitivity of y with respect to x_i can be derived by taking the partial derivative of f with respect to x_i as follows

$$S_i = \frac{\partial f}{\partial x_i}. \tag{4.3}$$

This simple example is typical of a *one variable at a time* approach (OAT), where one variable is evaluated at a time while the other variable are fixed to an assumed value. OAT is only appropriate for analyzing linear models since it does not take into account the neighbourhood of alternate assumptions as to where the other variables should be fixed [131, 132, 134]. Despite this limitation, OAT is still the most prevalent approach used in sensitivity analysis [134] - mainly due to its relative simplicity and ease of implementation.

More rigorous forms of uncertainty and sensitivity analysis require the simultaneous exploration of the entire space of uncertain input variables. This usually requires the application of Monte-Carlo sampling based approaches [134]. A simple example is one where M sample input sets are generated according to the estimated distribution of each input variable, $N(\langle x_i \rangle, \delta_i)$, $\forall i \in \{1, 2, .., N\}$

$$\mathbf{X}' = \begin{bmatrix} x_1^{(1)} & x_2^{(1)} & \dots & x_N^{(1)} \\ x_1^{(2)} & x_2^{(2)} & \dots & x_N^{(2)} \\ \vdots & \vdots & \vdots & \ddots \\ x_1^{(M)} & x_2^{(M)} & \dots & x_N^{(M)} \end{bmatrix}. \tag{4.4}$$

The model is evaluated with respect to each row of input values to produce a distribution of output values.

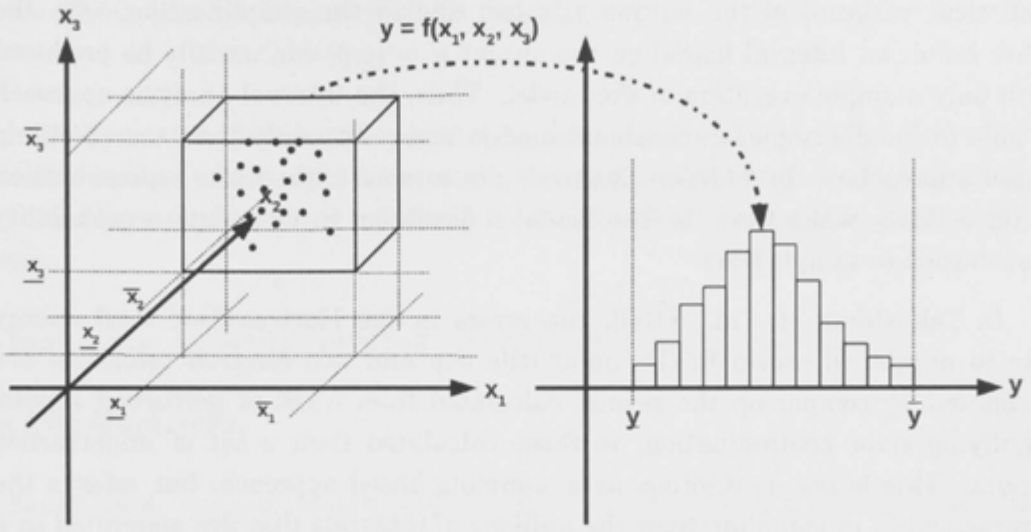


Figure 4.2: Sampling based estimate of the variance of $y = f(x_1, x_2, x_3)$

$$\mathbf{Y}' = \begin{bmatrix} y^{(1)} \\ y^{(2)} \\ \vdots \\ y^{(M)} \end{bmatrix}. \tag{4.5}$$

With a sufficiently large sample, it is possible to accurately predict the variance of the output as a result of variance in the input. Figure 4.2 illustrates the implementation of this approach in a three dimensional model $y = f(x_1, x_2, x_3)$ where x_1, x_2, and x_3 is uniformly distributed in $[\underline{x_1}, \overline{x_1}] \times [\underline{x_2}, \overline{x_2}] \times [\underline{x_3}, \overline{x_3}]$. A similar sampling based approach is used in global sensitivity analysis, albeit with the added complexity of having to apportion the variations to individual input variables or groups of input variables.

The sampling based approach is not without its limitations. There are high computational costs involved when evaluating models with large numbers of uncertain input variables due to the *curse of dimensionality*. Furthermore, when information about input uncertainty is sparse, naively assuming a particular probability distribution may lead to a critical underestimation of actual output variability [34].

The interval analysis approach can be seen as a simplification of the sampling based approach: where the error distribution of each uncertain input variable distribution $x_i = N(\langle x_i \rangle, \delta_i)$ is replaced by a simple interval bound $x_i = [\underline{x_i}, \overline{x_i}]$.

Like the sampling approach, interval analysis can intrinsically account for the entire range of uncertainty in the output. However, information about the

statistical variance of the output y is lost due to the simplification. On the other hand, an interval bound on the output $y = [\underline{y}, \overline{y}]$ can usually be produced with only a single execution of the model. Thus, the interval analysis approach is able to handle higher-dimensional models more efficiently than the sampling based approaches. In addition, intervals are a more appropriate representation of uncertainty when there is insufficient information to formulate a probability distribution to sample from.

In Takashima et. al. [150], the errors in the Hartree-Fock total energy due to numerical errors in the input (the one and two electron integrals) are estimated by comparing the results calculated from a set of perturbed inputs, signifying error contamination, to those calculated from a set of unperturbed inputs. This is not as rigorous as a sampling based approach, but reflects the impracticality in sampling from the millions of integrals that are generated in a typical Hartree-Fock calculation.

In summary, the interval analysis approach is appropriate for studying numerical errors in Hartree-Fock computation for the following reasons:

1. It provides a rigorous worst-case estimate of the range of output variability due to uncertainties introduced throughout the course of the calculation.
2. It is robust in that it can represent many different sources of uncertainty, such as those due to rounding error, truncation error, numerical instability etc.
3. The output variability can be bounded without having to sample the space of input variability.
4. There are sources of uncertainty, such as rounding error or truncation error, which are difficult to represent as an error distribution to sample from. An interval representation is often the only appropriate representation.
5. Unlike sampling based approaches, an interval model usually requires only a single execution in order to derive a bound, making it ideal for studying high dimensional models, such as the Hartree-Fock model.

4.3 Experimental Platform

All experiments were conducted on two Sun Microsystems SPARC platforms,

- ***Alcatraz*** - A V1280 UltraSPARCIIICu System with 12 $\times$ 900 MHz cores.

- ***Mavericks*** - A UltraSPARC T2 System with 8 $\times$ 1.2 MHz cores.

Both use the Solaris 10 operating system [143]. All programs were implemented in C++ and the SPARC assembly language, and compiled using SunStudio 11 compilers at patch level 20060426 [146]. For ***Alcatraz*** the basic build options were,

```
-fast -xia -xtarget=ultra3 -xarch=v9b -xlic_lib=sunperf
```

and for ***Mavericks***,

```
-fast -m64 -xtarget=ultraT2 -xia -I. -xlic_lib=sunperf -dalign
```

where the `-xia` flag enables the interval arithmetic library. OpenMP was used to parallelize the evaluation of two-electron integrals - this is enabled using the `-xopenmp` compiler flag.

4.4 Numerical Errors in Evaluating $F_m(T)$

Since the reduced incomplete gamma function or $F_m(T)$ forms the basis from which one and two-electrons are constructed, the numerical errors in $F_m(T)$ are likely to propagate throughout a Hartree-Fock calculation. It is therefore important to consider approaches which minimize the errors in $F_m(T)$. The purpose of this section is to use interval analysis to evaluate and compare the accuracy of various approaches for computing $F_m(T)$, taking into account factors including i) the type of polynomial interpolation scheme, ii) the degree of interpolation, iii) scheme in which the polynomials are evaluated, and iv) the application of accurate summation and dot product techniques.

These experiments follow from the work by Takashima [149] who evaluated the numerical accuracy of various $F_m(T)$ polynomial interpolation schemes. Their work focused on the effect of rounding errors when using double precision computation; this was measured by taking the mean squared difference between the $F_m(T)$ values calculated using double precision (52-bit mantissa) and $F_m(T)$ values calculated using extended double precision (64-bit mantissa) arithmetic. Takashima did not, however, consider the effect of errors as a result of Shavitt's series evaluation, or the impact of truncation error.

The Shavitt series evaluation methods to be compared are detailed in Table 4.1, while the polynomial interpolation schemes considered are outlined in Table

4.2. The labels **Taylor** and **Cheby** refer, respectively, to Taylor and Chebyshev interpolation, based on the naming conventions used in [149]. The term *Horner on* $(x_i - t)$ indicates that Horner's method was used to evaluate polynomials of $(x_i - t)$. The methods used to calculate error bounds follows directly from the interval extensions introduced in Section 3.3.

Rec	Recursive Summation (Algorithm 14)
Comp	Compensated summation (Algorithm 15)
Casc	Cascaded Accumulated Summation

Table 4.1: Summary of Shavitt series evaluation approaches.

Interpolation Scheme	n	Polynomial Eval	Equation
TaylorA	3	Horner on $(x_i - T)$	(3.19)
TaylorB	6	Horner on $(x_i - T)$	(3.19)
ChebyA	3	Horner on T	(3.26)
ChebyB	3	Clenshaw's on $(x_i - T)$	(3.26)
ChebyC	6	Clenshaw's on $(x_i - T)$	(3.26)

Table 4.2: Summary of polynomial interpolation schemes.

A range of commonly encountered T and m values are considered. In Shavitt series evaluation and polynomial interpolation, $F_m(T)$ values are evaluated for T from 0 to T_F at increments of 0.125 and m from 0 to 12 at increments of 1. The values of T_F for different m values are summarized in Table 3.1. In experiments involving the asymptotic expression, $F_m(T)$ values are evaluated for T from T_F to $1,000$ at increments of 1, and m from 0 to 12 at increments of 1. The number of terms (n) evaluated in Shavitt's series, and the granularity of the interpolation table are dictated by the truncation error threshold, which is set at the machine precision (IEEE double precision).

Since all numerical errors are contained within the interval extension of $F_m(T)$, the worst case numerical errors can be measured by evaluating the width of the intervals returned. The relative numerical error is given by the following expression

$$Err = \frac{w(F_m(T^I))}{m(F_m^{ref}(T^I))} \tag{4.6}$$

where $F_m(T^I)$ is the interval error bound of $F_m(T)$, and $T^I \in I(\mathbb{R})$ is a degenerate interval. $F_m^{ref}(T^I) \in I(\mathbb{R})$ is a reference interval obtained by taking the intersection

of all interval bounds of $F_m(T)$ computed during the course of the experiments. The average relative numerical errors are measured by taking the root mean squared averages of all relative numerical errors. When no confusion is likely to arise, all references to errors are taken to refer to the worst case error.

The calculation of $F_m(T)$ may involve evaluating i) Shavitt's series, ii) Taylor or Chebyshev polynomial interpolants, and/or iii) the Asymptotic expression. Thus, we proceed by examining each evaluation separately and then in combination.

The average relative numerical errors for Shavitt series evaluation using Rec, Comp, and Casc are shown on Table 4.3. The results show an accuracy of around 14 to 15 decimal digits of precision depending on which approach was used. Comp and Casc are able to compute $F_m(T)$ values that are around two to three times more precise than those computed using Rec. On the other hand, there are only slight differences between Comp and Casc, despite Casc being significantly more expensive to compute. This suggests that, out of the methods considered, compensated summation is the best method to use when performing Shavitt's series evaluation when taking into account both performance and precision.

	m			
Appr	0	4	8	12
Rec	1.3e-14	1.5e-14	1.7e-14	1.8e-14
Comp	3.5e-15	4.4e-15	4.8e-15	5.2e-15
Casc	3.4e-15	4.2e-15	4.7e-15	5.1e-15

Table 4.3: The average numerical error of $F_m^{Shav}(T)$ for various Shavitt series evaluation approaches.

Table 4.3 also shows that numerical error increases with m. This effect can be explained by observing Figure 4.3 which shows that the numerical errors also increased with T. This particular result was expected since more terms of Shavitt's series are required to calculate $F_m(T)$ values for larger T. When m increases, T_F will also increase, thus requiring larger T values to be evaluated.

It is possible to perform summations that are accurate to almost machine precision [122, 116], however this is not the case here since each term of Shavitt's series requires a floating division operation, which is not handled by either compensated summation or cascaded accumulated summation [121]. For this particular case, a possible solution would be to compute the numerator and denominator of each term separately using the numerator and denominator of the previous term. Although, a division would still be required to calculate each term, the rounding error stemming from this would not be propagated to the next term.

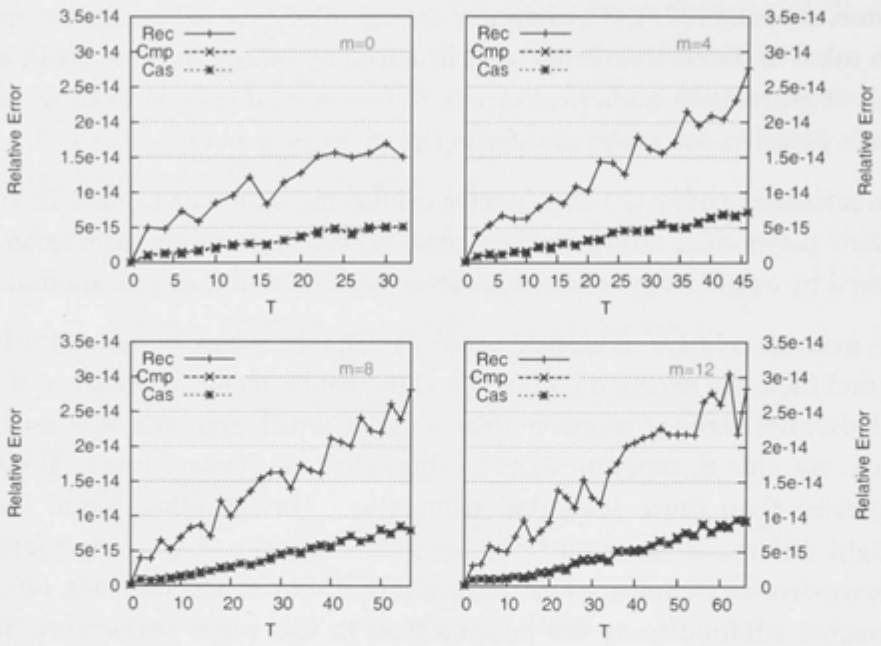


Figure 4.3: Relative numerical error of $F_m^{Shav}(t)$, for t from 0 to T_F, for various Shavitt series evaluation approaches.

Some provision would, however, have to be made for overflow in the value of the numerator and denominator in this procedure.

The average relative numerical error for each polynomial interpolation scheme is given in Table 4.4. These schemes may use either the Rec (in bold) or the Comp Shavitt series evaluation methods to generate interpolation points. The rows labelled N-P (in italics) involve test cases where numerical errors from Shavitt series evaluation are *not propagated* on to the final result. This therefore gives a final interval that contains only the numerical errors as a result of polynomial interpolation.

Table 4.4 shows the $F_m(T)$ values in the N-P case to be accurate to the order of 15 decimal digits, except for those computed using ChebyA which were only accurate to the order of 12 decimal digits for larger m values. This agrees with the earlier findings of Takashima [149], who showed that the higher than usual errors in ChebyA are caused by cancellation when evaluating Horner's method over T (Equation 3.28).

Aside from ChebyA, there are only minor variations between the other interpolation schemes. This could be due in part to the fact that increasing the degree of expansion from 3 to 6, as in ChebyB and ChebyC, introduces only a few additional operations that do not substantially increase rounding errors. It is also interesting to note that there is very little differences between Taylor and

Scheme		m = 0	m = 4	m = 8	m = 12
ChebyA	N-P	*6.6e-15*	*3.0e-13*	*1.5e-12*	*4.0e-12*
	Rec	**8.3e-14**	**4.3e-12**	**2.2e-11**	**6.7e-11**
	Comp	3.2e-14	1.6e-12	7.9e-12	2.3e-11
ChebyB	N-P	*8.7e-16*	*1.5e-15*	*2.0e-15*	*2.6e-15*
	Rec	**1.4e-14**	**1.7e-14**	**1.9e-14**	**2.1e-14**
	Comp	5.7e-15	7.2e-15	8.1e-15	8.9e-15
ChebyC	N-P	*1.2e-15*	*1.8e-15*	*2.4e-15*	*2.9e-15*
	Rec	**1.5e-14**	**1.8e-14**	**2.0e-14**	**2.1e-14**
	Comp	7.1e-15	8.5e-15	9.3e-15	1.0e-14
TaylorA	N-P	*4.0e-16*	*1.0e-15*	*1.6e-15*	*2.1e-15*
	Rec	**1.4e-14**	**1.7e-14**	**1.8e-14**	**2.0e-14**
	Comp	4.8e-15	6.2e-15	7.1e-15	8.0e-15
TaylorB	N-P	*4.1e-16*	*1.0e-15*	*1.6e-15*	*2.1e-15*
	Rec	**1.4e-14**	**1.7e-14**	**1.9e-14**	**2.0e-14**
	Comp	5.6e-15	7.0e-15	7.9e-15	8.7e-15

Table 4.4: The average relative numerical errors of $F_m(T)$ for various interpolation schemes.

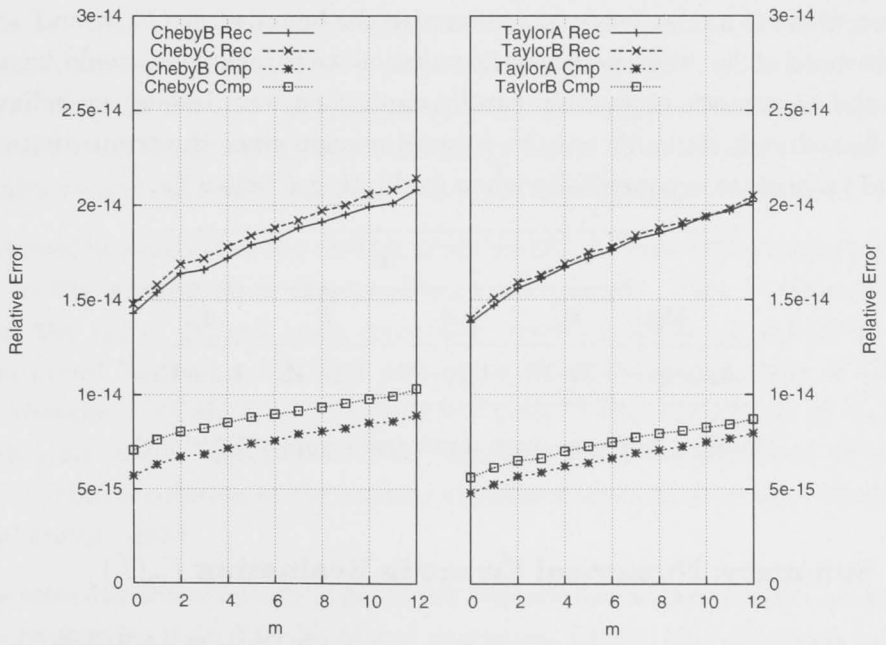


Figure 4.4: Average relative numerical errors of $F_m(T)$, for m from 0 to 12, for various interpolation schemes. The results for ChebyA are excluded.

Chebyshev interpolation in this regard.

When the error bounds from Shavitt series evaluation are propagated, we see clear differences between the schemes that use Rec and the schemes that use Comp. The polynomial interpolation schemes that used Rec produced results that are precise to around 14 decimal digits, whereas those that used Comp are precise to around 14 to 15 decimal digits - almost one order of magnitude difference. This is consistent with the results reported in Table 4.3, which also suggests that the accuracy of $F_m(T)$ is significantly influenced by the choice of Shavitt series evaluation method. While, on the other hand, it appears that the choice of interpolation scheme again has very little influence except when ChebyA is used.

The average relative numerical error in evaluating the asymptotic expression is outlined in Table 4.5. The accuracy of all reported results are in the order of 15 decimal digits, or around an order of magnitude more accurate than those computed using polynomial interpolation. This is likely to be because the asymptotic expression requires fewer operations to compute, therefore making it less likely for rounding errors to accumulate. No significant variation in error was observed for different values of T and m, except when $m = 0$ where the results were the most accurate as even less operations are required.

This result implies that $F_m(T)$ can be evaluated more accurately when T is greater than T_F compared to when T is less than T_F. One may naturally enquire then since there is a relatively steep change in the bound of $F_m(T)$ when T is in the neighbourhood of T_F , then perhaps the value of the threshold T_F could be reduced so that the asymptotic expression can be applied earlier. Indeed we believe that T_F can be reduced, although only by a small margin since the truncation error of $F_m^{Asymp}(T)$ increases exponentially when evaluating T below T_F.

	m			
Alg	0	4	8	12
Asymp	4.3e-16	1.1e-15	1.1e-15	1.1e-15

Table 4.5: The average numerical errors of $F_m^{Asymp}(t)$.

4.4.1 Summary: Numerical Errors in Evaluating $F_m(T)$

The reduced incomplete gamma function or $F_m(T)$ is a fundamental quantity required in Hartree-Fock computations. This results in this section demonstrate that $F_m(T)$ can be evaluated accurately, depending on factors such as i) the values of m and T, ii) the value of T in relation to the threshold value of T_F, and iii) the approach used in evaluating the Shavitt series. By using the certain approaches,

it was possible to guarantee that relative numerical errors in will not exceed $\approx 1 \times 10^{-14}$ or affect more than two decimal digits of precision. The most effective approach, in terms of precision and efficiency, is to use compensated summation to evaluate the Shavitt series. The choice of polynomial interpolation scheme, however, was not an important factor, except in the case of ChebyA, which is affected by cancellation errors.

4.5 Numerical Errors in Evaluating the Hartree-Fock Total Energy

Supposing that E^I_{HF} is the interval containing the Hartree-Fock total energy, then the relative numerical error of the total energy can be expressed as

$$Err = \frac{w(E^I_{HF})}{m(E^I_{HF})}$$

This is equivalent to dividing the width of E^I_{HF} by its midpoint. For clarity, we will often state the result in terms of *precision* instead of error, where $-log_{10}(err)$ is used to indicate the number of *decimal digits of precision* in the result. For instance, $err = 1 \times 10^{-13}$ implies 13 decimal digits of precision. If the computation is implemented in double precision there are a maximum of 16 decimal digits, so a relative error of 1×10^{-13} implies that 3 decimal digits have been loss due to numerical errors. Since we are evaluating E^I_{HF} using interval analysis techniques, the results should naturally be construed as an estimate of worst case errors.

It should be noted that the error bounds obtained in the following experiments relates to the ground state Hartree-Fock total energy. This is obtained using as input the set of ground state molecular orbital coefficients calculated using a conventional floating point SCF procedure. The interval bounds of the total energy therefore reflect errors propagated from i) the evaluation of $F_m(T)$, ii) the evaluation of one- and two-electron integrals, iii) the construction of the Fock matrix, iv) the calculation of the nuclear repulsion term, and iv) the calculation of the total energy itself.

The total electronic energy is generally regarded to be *chemically accurate* if its error is no greater than 0.01 KCal/mol or around 1.59×10^{-5} Hartrees [150]. As a consequence, larger systems with larger total energies will require more digits of precision in order to be chemically accurate. However, calculations on large molecules also require more operations to be performed so there is greater potential for accumulating errors. As interval arithmetic bounds the numerical errors it can be used to rigorously guarantee that chemical accuracy is achieved.

$(H_2O)_n$	N	3-21G	N	6-31G**	N	cc-pVDZ	N	6-31++G**	N	6-311G(2df,2pd)
2	26	12.63	50	12.26	50	12.14	62	11.98	130	11.48
3	39	12.30	75	11.92	75	11.76	93	11.63	195	11.12
4	52	12.08	100	11.69	100	11.57	124	11.41	260	10.88
5	65	11.87	125	11.48	125	11.31	155	11.19	325	10.56
6	78	11.70	150	11.29	150	11.16	186	11.01		
7	91	11.56	175	11.15	175	11.04	217	10.85		
8	104	11.41	200	11.00	200	10.85	248	10.69		
9	117	11.28	225	10.88	225	10.73	279	10.58		
10	130	11.17	250	10.77	250	10.64	310	10.46		
11	143	11.08	275	10.60						
12	156	11.00	300	10.53						
13	169	10.92	325	10.45						
14	182	10.84	350	10.37						
15	195	10.77								
16	208	10.71								
17	221	10.65								
18	234	10.60								
19	247	10.54								
20	260	10.50								

Table 4.6: Decimal digits of precision of E_{HF} for TIP4P water clusters of n water molecules computed using 3-21G, 6-31G**, cc-pVDZ, 6-31G++**, and 6-311G(2df,2pd) basis sets. N is the number of basis functions.

For the following experiments, it should be assumed that all integral values are evaluated using a natural interval extension of the Head-Gordon-Pople (HGP) algorithm, and all $F_m(T)$ values evaluated using sixth-degree Chebyshev polynomial interpolation with interpolation points generated using the compensated summation implementation of Shavitt's series. The latter settings were chosen because the results in Section 4.4 showed that they yielded tight bounds on $F_m(T)$.

4.5.1 Effect of Basis Set and System Size and Composition on the Total Energy

To study the effects of basis set and system size we evaluate water clusters that range from the near minimal 3-21G basis set to the extensive 6-311G(2df,2pd) basis set [137]. The water molecule conformations that is used correspond to the TIP4P water model which can be found in the *Cambridge Cluster Database* [157].

Results are shown in Table 4.6 for systems with total basis set sizes of up to 350 basis functions. As expected, increasing the number of atoms and/or the basis set size causes the number of decimal digits of precision to decrease. With the 3-21G basis set, going from 2 to 20 water molecules results in a loss of more than

2 significant figures in the total energy. In terms of chemical accuracy another significant figure is effectively lost as the total energy also increases by more than one order of magnitude in going from 2 to 20 water molecules. With 20 water molecules, and a total energy in the order of 10^3, 10.50 digits of decimal precision is still however sufficient to give total energies accurate to 10^{-5} Hartrees.

For the smaller systems, increasing the basis set size from 3-21G to 6-311G (2df,2pd) also results in a loss of more than one decimal digit of precision in E_{HF}. Comparing the 6-31G** and cc-pVDZ basis sets, both of which have the same number of contracted functions but achieved using different contraction schemes, reveals that the cc-pVDZ correlation consistent basis set gives slightly fewer digits of precision. Adding diffuse functions to the 6-31G** basis gives a slight further loss in precision. Interestingly the number of digits of precision appears to be relatively constant for a given total number of functions N regardless of whether this results from a large number of atoms or from a large basis set.

To explore the latter point further we plot in Figure 4.5 the relative errors from Table 4.6 as a function of N. A second-degree polynomial regression was used to fit the data for each basis set type. The R^2 (coefficient of determination) values of the regressions were found to be 0.9997, 0.9959, 0.9958, 0.9977, and 0.9841 for the 3-21G, 6-31G**, cc-pVDZ, 6-31++G, and 6-311G (2df,2pd) basis sets, respectively. This strongly suggests that a second-degree polynomial model is sufficient to explain the variation of numerical errors due to N. Linear regression was also considered, but was found not to yield as good a fit. The success of the second polynomial is likely to be related to the numerical complexity of the Hartree-Fock method. For example, in an N-term summation of $O(N)$ complexity, the growth in numerical errors is in the order $O(N)$, but also depends on the type of terms involved [46]. In the Hartree-Fock method the numerical complexity grows as $O(N^4)$. For real systems, however, larger problem sizes mean larger spatial extent, and since the Coulombic interaction decreases with distance it is not surprising to find that the numerical errors increase in a non-linear fashion, but at a rate that is less than quartic. Finally, aggregating together the results for all basis set types, and then applying regression, also leads to a slightly poorer fit; which, suggests that there are other determining factors besides N.

While the least precise results found in Table 4.6 were still well within chemical accuracy, loss of precision is likely to become an issue with larger clusters and/or with larger basis sets. Given its ability to fit existing data, the polynomial regression models shown in Figure 4.5 can also be used to predict errors for larger values of N. This is shown in Figure 4.6 for the four basis sets used. The plot indicates that relative numerical errors are generally of the order of 1×10^{-8} when $N = 10,000$, with the specific values given in Table 4.7.

3-21G	6-31G**	cc-pVDZ	6-31++G**	6-311G(2df,2pd)
5.58×10^{-8}	4.64×10^{-8}	4.39×10^{-8}	4.40×10^{-8}	3.35×10^{-8}

Table 4.7: Predicted relative numerical error at $N = 10,000$ for TIP4P water clusters computed using different types of basis sets.

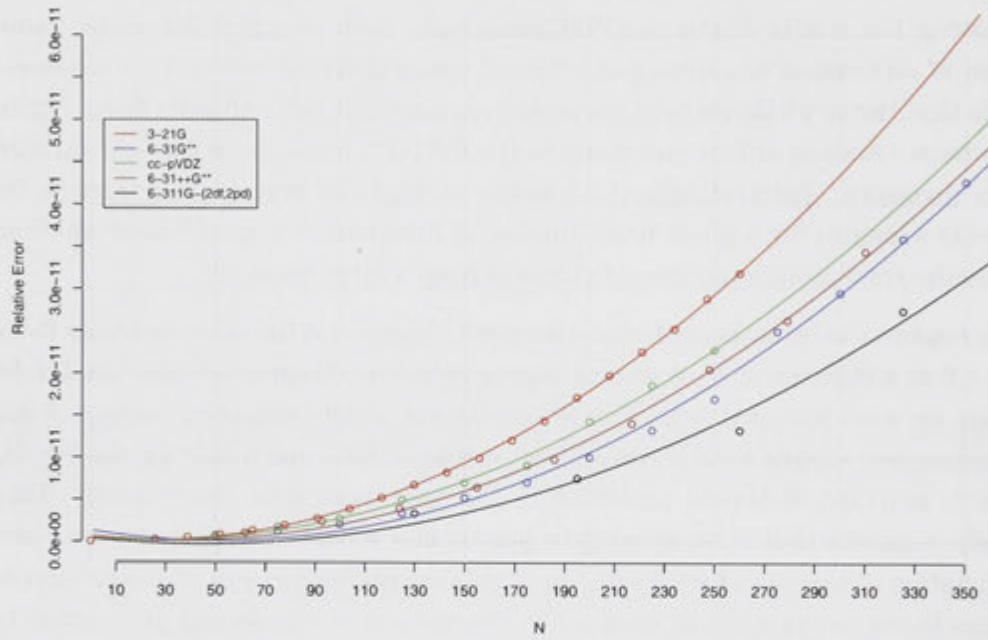


Figure 4.5: Relative error of E_{HF} for water clusters fitted against the number of basis functions N, using second degree polynomial regression.

The largest system considered in the experiment above was $(H_2O)_{15}$ 6-31G** with 350 basis functions. The system sizes that can be studied is limited by the performance of the interval code. Interval computation is inherently slower than floating point computation due to the additional storage requirement, and the additional number of floating operations required to achieve a single interval arithmetic operation (Equation 2.58). The *Interval Hartree-Fock* code is also experimental in nature, and therefore not optimized to the same degree as better established floating point codes. Moreover, in order to guarantee rigorous bounds on numerical errors, time saving heuristics such as two-electron integral screening cannot be applied as with floating point codes. In fact all $O(N^4)$ two-electron integrals were calculated using interval arithmetic in our implementation. In actual experiments, the interval Hartree-Fock code was observed to be around 10 to 25 times slower than its floating point equivalent. The ratio of execution times

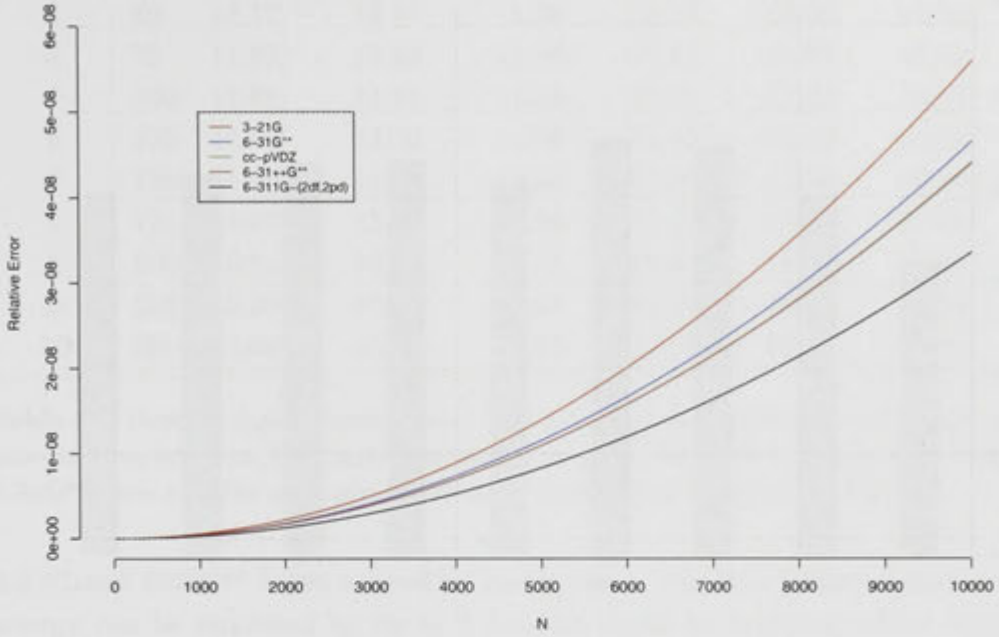


Figure 4.6: Predicted Relative error of E_{HF} for water clusters of size up to $N = 10,000$ using second degree polynomial regression models derived from existing data.

required to compute the entire set of two-electron integrals for water clusters of 6-31G** using floating point and interval arithmetic is summarized in Figure 4.7.

4.5.2 Effect of $F_m(T)$ Interpolation Scheme on the Total Energy

The following experiment investigates the impact that the choice of $F_m(T)$ interpolation scheme has on the accuracy of the Hartree-Fock total energy. The interpolation schemes considered include TaylorA, TaylorB, ChebyA, ChebyB, and ChebyC (see Table 4.2). Series evaluation is performed using compensated summation in all cases, except for ChebyA-Rec where recursive summation is used (together with ChebyA). Therefore, ChebyA-Rec represents the worst possible configuration for computing $F_m(T)$ according to the results of Section 4.4.

Table 4.8 compares the precision of 6-31G** water cluster total energies computed based on the different $F_m(T)$ interpolation schemes. The most precise result for each cluster is highlighted in bold. The energies computed using ChebyA and ChebyA-Rec were clearly the least precise. This was not surprising since the results of Section 4.4 show that ChebyA was also the least precise scheme for computing $F_m(T)$ values. In the majority of the cases, ChebyA is around 0.5

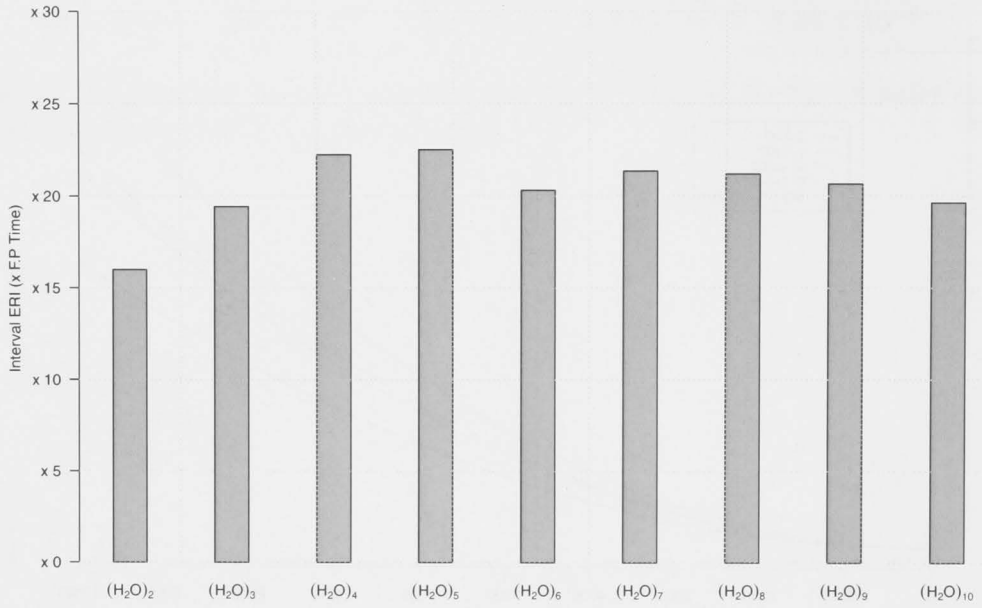


Figure 4.7: Execution times for ERI evaluation performed in interval arithmetic for $(H_2O)_n$/6-31G**, expressed as a ratio of the execution times for the equivalent evaluation performed in floating point arithmetic.

to 1 decimal digits less precise than the other interpolation schemes, although the differences appear to decrease with the problem size, possibly because the $F_m(T)$ errors have been superseded by other sources of error. ChebyC was the most precise interpolation scheme, although there were only marginal differences between it and TaylorA, TaylorB, and ChebyB.

4.5.3 Calculating the Total Energy using Compensated Summation and Cascaded Accumulated Summation

In Section 4.4, the accuracy of Shavitt series evaluation was improved significantly with the application of compensated summation and cascaded accumulated summation techniques. The following experiment investigates whether similar improvements can be obtained when these techniques are applied to evaluate the Fock matrix and the total energy.

Table 4.9 compares the numerical precision of compensated summation (Comp) and cascaded accumulated summation (Casc) with the conventional approach denoted as Control. The system considered are TIP4P water clusters using the

$(H_2O)_n$	N	Taylor A	Taylor B	ChebyA	ChebyB	ChebyC	ChebyA-Rec
2	50	12.17	12.17	11.55	12.17	**12.26**	11.33
3	75	11.82	11.82	11.38	11.82	**11.92**	11.23
4	100	11.61	11.61	11.23	11.61	**11.69**	11.08
5	125	11.39	11.39	11.00	11.39	**11.48**	10.83
6	150	11.22	11.22	10.85	11.22	**11.29**	10.68
7	175	11.07	11.07	10.84	11.07	**11.15**	10.72
8	200	10.91	10.91	10.65	10.91	**11.00**	10.50
9	225	10.80	10.80	10.64	10.80	**10.88**	10.55
10	250	10.69	10.69	10.55	10.69	**10.77**	10.48

Table 4.8: Decimal digits of precision of E_{HF} computed using different $F_m(T)$ polynomial interpolation schemes. The total energies are computed for TIP4P water clusters using the 6-31G** basis set. The most precise result for each cluster is highlighted in bold.

3-21G and 6-31G** basis sets. The results show that the precision of the total energy can be improved by up to 2 decimal digits by applying either Comp or Casc. More significantly, the improvement obtained over the convention approach appears to increase with the problem size. For example, in $(H_2O)_{15}$ / 3-21G the compensated summation approach is accurate up to 12.50 decimal digits, whereas the conventional approach is only accurate up to 10.77 decimal digits. This indicates almost two orders of magnitude more precision.

The improvements observed as the problem size increases could be related to the $O(N^4)$ complexity of the problem. That is, as the problem size increases, a larger proportion of the computation is performed using accurate summation techniques, therefore the precision should continue to increase relative to the conventional recursive summation code.

On the other hand, the results also show no significant differences between Comp and Casc in any of the systems considered; which, suggests that the added expense of performing cascaded accumulated summation is unnecessary for this application.

When the compensated summation results are projected towards $N = 10,000$ using the second-degree polynomial regression model, the relative error for 3-21G and 6-31G** was found to be 2.50×10^{-10} and 5.27×10^{-11}, respectively. This is two to three orders of magnitude more precise than the projected error for the conventional approach shown in Table 4.7.

	3-21G					6-31G**				
$(H_2O)_n$	N	Control	Comp	Casc	Imp	N	Control	Comp	Casc	Imp
2	26	12.63	13.31	13.31	*+0.68*	50	12.26	13.14	13.14	*+0.88*
3	39	12.30	13.20	13.20	*+0.90*	75	11.92	13.01	13.01	*+1.09*
4	52	12.08	13.12	13.13	*+1.05*	100	11.69	12.92	12.92	*+1.23*
5	65	11.87	13.07	13.07	*+1.20*	125	11.48	12.87	12.87	*+1.39*
6	78	11.70	12.99	13.00	*+1.30*	150	11.29	12.77	12.77	*+1.48*
7	91	11.56	12.92	12.92	*+1.36*	175	11.15	12.72	12.72	*+1.57*
8	104	11.41	12.87	12.87	*+1.46*	200	11.00	12.67	12.67	*+1.67*
9	117	11.28	12.82	12.82	*+1.54*	225	10.88	12.62	12.62	*+1.74*
10	130	11.17	12.78	12.78	*+1.61*	250	10.77	12.58	12.58	*+1.81*
11	143	11.08	12.72	12.72	*+1.64*					
12	156	11.00	12.68	12.68	*+1.68*					
13	169	10.92	12.62	12.62	*+1.70*					
14	182	10.84	12.57	12.57	*+1.73*					
15	195	10.77	12.54	12.54	*+1.75*					

Table 4.9: Decimal digits of precision of E_{HF} computed using i) the conventional approach (Control), ii) compensated summation (Comp), and iii) cascaded accumulated summation (Casc). The largest improvement (Imp) obtained over the conventional approach is also indicated. The total energies are computed for TIP4P water clusters using the 3-21G and 6-31G** basis sets.

4.6 Effect of Two-Electron Integral Screening on the Total Energy

For reasons of efficiency, it is customary in HF calculations to pre-screen the ERIs and only compute those with a magnitude greater than some threshold. The most widely used screening technique is based on the *Schwartz* test [6]. This sets the following upper bound for value of an ERI:

$$|(\chi_a\chi_b|\chi_c\chi_d)| \leq \kappa_{ab}\kappa_{cd}, \quad \text{where } \kappa_{ab} = \sqrt{(\chi_a\chi_b|\chi_a\chi_b)}. \tag{4.7}$$

Use of Schwartz screening requires pre-computation of two centre ERIs of the form $(\chi_a\chi_b|\chi_a\chi_b)$. Before computing a specific integral the value of $\kappa_{ab}\kappa_{cd}$ is computed and compared against the *screening threshold* τ. If $\kappa_{ab}\kappa_{cd} > \tau$ the actual integral is evaluated, otherwise it is neglected. For efficiency, Schwartz screening is usually performed for batches of integrals, not for individual integrals [6].

In Schwartz screening, choosing a value for τ represents a trade-off between speed and accuracy: set too high and the accuracy will be affected, set too low and the time to solution increases but with insignificant improvement in numerical accuracy. Interval arithmetic can be used to quantify the numerical error that

System	N	No. ERI ($\times 10^6$)	τ			
			ϵ	1×10^{-12}	1×10^{-8}	1×10^{-4}
$(H_2O)_4$	100	11.29	11.69	11.66 [17.98%]	10.25 [35.44%]	6.08 [63.93%]
Alanine	125	29.11	11.41	11.35 [6.87%]	11.19 [17.17%]	6.12 [45.86%]
Serine	140	44.77	11.30	11.26 [10.56%]	11.25 [22.32%]	6.30 [50.68%]
Cytosine	145	59.77	11.24	11.22 [14.20%]	9.92 [25.92%]	6.38 [50.73%]
$(H_2O)_8$	200	154.91	11.00	10.98 [35.13%]	9.94 [56.27%]	5.64 [80.05%]
$(H_2O)_{10}$	250	492.21	10.77	10.44 [49.60%]	9.98 [67.29%]	5.62 [83.99%]

Table 4.10: Decimal digits of precision of E_{HF} for different Schwartz screening thresholds. The entries inside the square brackets indicate the percentage of unique ERI that are screened with respect to each cut-off.

results from using different values of τ, and to determine, rigorously, the largest value of τ capable of maintaining chemical accuracy. The error arising from integral screening is estimated using the following expression

$$Err = \frac{\mathrm{mag}(E^I_{HF} - E^\tau_{HF})}{m(E^I_{HF})} \tag{4.8}$$

where E^I_{HF} is the interval bound of the total energy calculated without integral screening, E^τ_{HF} is the interval bound of the total energy calculated using an integral screening threshold of τ; $\mathrm{mag}(.)$ in this context represents the greatest difference between E^I_{HF} and E^τ_{HF} [1]

In Table 4.10, we show the number of digits of precision in E_{HF} for ERI screening thresholds ranging from the machine epsilon (ϵ) up to 1×10^{-4}. Five different systems are considered, including three water clusters and three amino acid systems, each computed using the 6-31G** basis set. The values in square brackets denote the percentage of unique ERIs that are screened away. [2]

The results show that when using screening at machine epsilon, E_{HF} is accurate to between 11 and 12 significant digits. When using a screening value of 1×10^{-12}, there is minimal effect on the number of significant digits in E_{HF}, but the number of computed integrals decreases by between 7% and 50%. Increasing the screening value to 1×10^{-8} results in a slightly larger loss of precision which is still within chemical accuracy. Further increasing the screening value to 1×10^{-4},

[1] It should be noted that this is a different metric to that used in [59]. In the latter publication, the errors from integral screening are evaluated by replacing each screened integral by $[-\kappa_{ab}\kappa_{cd}, \kappa_{ab}\kappa_{cd}]$. While this is a more efficient approach that does not require the computation of E^I_{HF}, it generally yields more pessimistic error bounds that the method used in this section.

[2] Note that in [59], ERIs of magnitude less than 10^{-16} were not included in this percentage thus giving different percentages to those reported here.

however, results in significantly larger loss of precision and values for E_{HF} that are precise to only 5 or 6 decimal digits of precision which is not accurate to within 10^{-5} Hartrees given the magnitude of the total energy.

4.6.1 Summary: Numerical Errors in Evaluating the Hartree-Fock Total Energy

In this section we investigated the errors in the Hartree-Fock total energy as a consequence of i) increasing basis set and system size, ii) the use of different $F_m(T)$ interpolation schemes, iii) the use of compensated summation and cascaded accumulated summation, and iv) the use of different integral screening thresholds.

The results highlight the gradual accumulation of numerical errors with increasing basis set and system sizes. which will eventually reach a stage where chemical accuracy can no longer be guaranteed. A significant finding was that the estimated growth in numerical errors can be accurately modelled by regressing the number of basis functions against the width of the interval bound E_{HF}. The best fit was achieved using second degree polynomials. Non-linear scaling such as this was expected since the computational complexity scales non-linearly with basis set size.

It was shown that the choice of $F_m(T)$ polynomial interpolation scheme has an appreciable effect on the precision of the total energy. The findings correspond with previous observations showing that $F_m(T)$ values computed using third-degree Chebyshev interpolation and Horners method (**ChebyA**) was the least precise.

The results also show that the numerical errors in the total energy can be reduced significantly when the Fock matrix and the total energy evaluation steps are performed using compensated summation or cascaded accumulated summation. Furthermore, the results also show that these improvements increase as the system size becomes larger, which at $N = 10,000$ may result in total energies that are approximately two to three orders of magnitude more precise than those obtained using the conventional approach.

Finally, these findings also support the generally accepted view that it is possible to use integral screening to significantly reduce the computational scaling of large HF computations, without sacrificing chemical accuracy.

4.7 The Graphics Processing Unit

In the previous section, we considered numerical errors in total energy evaluations, performed using algorithms designed to execute efficiently on a conventional CPU. The following sections investigate the consequences of adopting alternate algorithm designs that are optimized towards execution on specialized processor architectures; using Graphics Processing Units (GPU) as the main example.

The rapid growth in the gaming industry has led to increased development in specialized processors for graphics processing. A notable example in this domain is NVIDIA's Compute Unified Device Architecture (CUDA) enabled graphics processing units [114]. GPUs are primarily designed to execute highly data parallel tasks inherent to the graphics pipeline. A desirable by-product of this design focus is the capability to perform floating point operations at rates significantly faster than possible on a conventional CPU. For instance, the now three years old NVIDIA GTX280 GPU is capable of 933 GFLOP/s in single precision, whereas the more recent Intel i7 980XE CPU is only capable of 213 GFLOP/s in single precision. The potential performance and cost advantages that GPUs can provide has led to growing interest in their application to scientific computing.

A relatively large portion of a GPUs on-chip resources is dedicated to supporting a massive number of parallel execution units. While this type of design is capable of large theoretical peak performance, it comes at the expense of other features that one might normally expect from a CPU. For example, i) GPUs will often lack the control logic for sophisticated branch prediction, ii) most GPUs will have relatively limited fast memory storage, and iii) until recently GPUs were not even capable of performing double precision arithmetic - even now, GPU double precision performance still tends to lag significantly behind single precision performance. The main obstacle to the adoption of GPUs for scientific computing applications has therefore been the task of designing and implementing computational codes that work around these limitations.

To further elucidate the issues surrounding GPU computation and its implications to Hartree-Fock calculations - the following section will provide a brief background on the architecture of the recent CUDA-enabled NVIDIA graphics cards, which are fast becoming the platform of choice for many scientific computing applications.

4.7.1 NVIDIA's CUDA Framework

Until recently, the only way to execute non-graphics related applications on a GPU was to painstakingly translate each non-graphic procedure into a series of calls to the graphics API. The situation has since improved with the introduction of the Compute Unified Device Architecture (CUDA), a parallel programming framework designed for (higher-end) NVIDIA graphics cards [114]. In brief, CUDA provides an interface for code written in variants of standard programming languages, including C and FORTRAN, to uncomplicatedly execute on a GPU, very much as if it was a conventional CPU.

Sections of code that are designated to be executed on the GPU are known as *kernels*, whereas the rest, assumed to be executed on the CPU, is referred to as the *host* code . A kernel is typically executed on multiple threads arranged into a *grid* of *thread blocks*. Each thread block may consist of up to 768, 1,024, or 1,536 threads depending on the particular model of the GPU. Although the dimensions of the grid and the size of each thread block can be specified arbitrarily, a grid must contain hundreds or even thousands of threads to take full advantage of GPU resources.

Some elements of the grid are physically mapped on to the GPUs execution units. Each thread block in the grid is mapped on to one of several *stream multiprocessor* (SM) on the GPU. An NVIDIA GTX280, for example, consists of 32 SMs. A typical SM consists of eight execution cores known as *stream processors* (SP). These are not cores in the same sense as cores on a multi-core CPU - they are considerably more primitive and are utilized much like SIMD units. This is underscored by the fact that each SP within the same SM can only execute the same instruction at each cycle (or simply stall). Each SP typically consists of an Arithmetic-Logic Unit (ALU) and a Floating Point Unit (FPU), both capable of executing 32-bit integer and floating point operations, respectively. Until recently there were no execution units for 64-bit double-precision floating point arithmetic. In recent NVIDIA GPUs, 64-bit ALUs and FPUs are now available, but are shared between SPs; meaning that double precision performance still lags considerably behind single precision performance.

All threads within the same thread block can potentially execute concurrently within the same SM. However, in practice, the limited number of SPs means that only a smaller subset of the thread block known as the *warp* (32-threads) may execute on the SM at any one time. Some scheduling is therefore required to organize the execution of the warps. Being able to swap warps in and out of an SM allows the program to hide memory access latency. For example, a warp that is stalled on a memory access can be swapped out for another warp that is ready to execute. Unlike a typical SIMD execution unit, the execution path of individual

threads within the same thread block may *diverge*. However, given the limitations of the SP, the execution of one group of threads in the same warp must be stalled while another group executes a different path. This poses significant performance issues in code where branching instructions such as if-statements and loops are necessary.

The memory hierarchy of a CUDA-enabled GPU consists of five main elements: *registers*, *shared memory*, *local memory*, *global memory*, and *constant and texture memory*. Registers, shared memory, and local memory are only visible to individual threads, or threads within the same thread block, whereas global memory and constant and texture memory are visible to all threads within the same grid.

Registers, and shared memory are fast, on-chip memory physically located on each SM. Like all fast memory they have relatively limited capacity compared to slower memory such as global memory. Registers are only visible to a thread, and are used primarily to store scalar variables. Shared memory is visible to all threads within the same thread block, and is utilized in a similar manner to the data cache on the CPU, i.e. to store the most frequently used program data. The main difference between shared memory and a data cache is that the programmer can directly specify what is stored in shared memory.

Register and shared memory usage affects the number of thread blocks that can be concurrently mapped onto an SM. For example, if the shared memory capacity of each SM is 16KB, and each thread block uses 2KB, then no more than 8 thread blocks can be mapped on to an SM at the same time. Allocating too much shared memory or using too many registers, therefore reduces the degree of concurrency, and by implication, reduces the ability of the program to hide memory access latency.

Local memory, global memory, and texture and constant memory all reside in slow, but large, off-chip memory. For example, the NVIDIA GTX280 has almost 1GB of global memory, compared to only 16KB of shared memory on each SM.

Local memory is used primarily to store per-thread static arrays, while, due to its size, global memory acts as the staging point for all the program data that the kernel requires. Although global memory accesses are typically un-cached, except on some recent GPUs, simultaneous accesses to consecutive blocks of memory can potentially be coalesced into a single access. Thus, global memory accesses should be optimized, in particular, for spatial locality. Texture memory and constant memory provide cached, read-only storage. When the data exists in cache, the access time is comparable to that of fast memory. However, the cache sizes available for per multiprocessor is only 6 KB to 8 KB for texture memory, and 8KB for constant memory.

4.8 Numerical Errors in GPU Computation

It was not long after the first applications of GPUs to scientific computing that researchers began to consider its potential to accelerate electronic structure calculations. The evaluation of the two-electron repulsion integrals (ERI) in particular was identified as one of the most critical areas of application. This was due to the significance of the ERIs, and the relative ease in parallelizing its evaluation. The first published work that considered ERI computation using GPUs was by Yasuda in 2008 [162]. This was followed up by broader research work conducted by Ufimtsev and Martinez [152, 153, 154], who developed code for performing Hartree-Fock SCF calculations, and geometry optimizations on a GPU.

In principle, all ERIs can be evaluated in parallel. However, the efficient evaluation of integrals usually requires the pre-computation of intermediate data such as interpolation tables, intermediate integral values, and program drivers. Since fast memory is especially limited on GPUs, the *Rys quadrature* method has so far been the preferred approach due to its small memory footprint relative to other recursive relation schemes [162, 152]. Ufimtsev and Martinez [152] worked around the warp divergence problem by implementing separate, largely sequential, evaluation kernels for each type of integral, i.e. $(ss|ss)$, $(ps|ss)$, $(pp|ss)$, etc.

Ufimtsev and Martinez [152] also evaluated the performance of several different integral to thread mappings, including i) one thread block to compute one contracted integral, ii) one thread to compute one contracted integral, and iii) one thread to compute one primitive integral.

While considerable effort have been invested in developing efficient Hartree-Fock code for GPUs, less attention has been given to consider how accurate these methods are. An obvious question is whether single precision calculations performed on a GPU is sufficient to provide chemical accuracy. Another lesser explored question relates to whether other error tolerances can also be relaxed in order to accelerate GPU computation without affecting chemical accuracy.

With these questions in consideration, the following experiments will use *Interval Hartree-Fock* to investigate i) the accuracy of results obtained using single precision, and whether a mix of double precision and single precision offers a better compromise in terms of accuracy and efficiency, and ii) to what extent can the error tolerances of the $F_m(T)$ interpolation tables be relaxed without compromising on chemical accuracy.

It is important to emphasise that all the results given here are obtained using *Interval Hartree-Fock* executed on a conventional CPU; that is, we are using the interval code to pose "what-if" questions without needing to have access to

hardware that supports that "what-if" scenario.

4.8.1 Effect of Using Mixed Arithmetic Precision on the Total Energy

While general purpose CPUs have traditionally provided strong double precision performance, this has not always been the case for GPUs and other specialized processors. For example the NVIDIA GTX280 GPU is capable of 933 GFLOP/s in single precision, but only 78 GFLOP/s in double precision. Similarly the CellBE PowerXCell 8i is capable of 230.4 GLOP/s in single precision, but only 108.8 GFLOP/s in double precision.

To early users it was clear that, in order to fully harness the potential of these systems, single precision arithmetic must be prioritized over double precision arithmetic when possible. Yasuda [162] sought to address this issue by partitioning integrals based on their Schwarz upper bounds; computing all integrals below some bound (λ_{GPU}) in single precision on the GPU, while other integrals greater than λ_{GPU} were calculated in double precision on the host CPU. The larger the value of λ_{GPU}, the larger the proportion of ERIs that can be calculated in single precision, and the faster the overall computation, but the lower the overall accuracy of the result. Yasuda showed that evaluating integrals entirely in single precision was not accurate enough for production runs.

Ufimtsev [152], while accepting Yasuda's approach, questioned whether mixing different floating point precisions was worthwhile given the arrival of double precision capable GPUs designed specifically for scientific computing. It can be argued, however, that GPUs first emerged as a cost effective solution not for scientific computing, but for an entirely different market segment where double precision is not important. This market segment is considerably larger than for scientific computing and will continue to only require single precision floating point and integer arithmetic for the foreseeable future. Thus the demands of scientific computing are always going to come second in driving GPU innovation. However, and as discussed in the introduction of this chapter, there are a number of other factors that motivate the use of short data types. For example, using less precision reduces storage requirements, and reduces the movement of data between different memory levels.

Interval analysis can be used to measure the numerical errors in E_{HF} due to different thresholds for λ_{GPU}. This differs from previously published work that simply compares single precision results to double precision results in that the intervals also bound the interactions with other sources of errors, not only those due to the value of λ_{GPU}. To simulate the application of the threshold,

			λ_{GPU}					
$(H_2O)_n$	N	D.P	1×10^{-12}	1×10^{-8}	1×10^{-4}	1×10^{-2}	$1 \times 10^{+0}$	S.P
2	50	12.26	12.26 [3.92%]	12.26 [13.92%]	10.81 [38.66%]	8.52 [60.17%]	6.15 [90.74%]	6.02
3	75	11.92	11.92 [6.99%]	11.92 [20.87%]	10.36 [51.69%]	8.17 [74.02%]	6.07 [90.81%]	5.95
4	100	11.69	11.68 [17.98%]	11.68 [35.44%]	10.56 [63.93%]	8.29 [80.07%]	6.03 [90.82%]	5.91
5	125	11.48	11.46 [34.54%]	11.46 [52.73%]	10.39 [74.61%]	8.11 [84.33%]	6.01 [90.82%]	5.88
6	150	11.29	11.20 [26.49%]	11.20 [46.15%]	10.12 [73.93%]	7.93 [85.59%]	5.97 [90.83%]	5.84
7	175	11.15	11.05 [32.31%]	11.05 [53.32%]	10.14 [77.75%]	7.94 [86.87%]	5.94 [90.83%]	5.82
8	200	11.00	10.98 [35.13%]	10.90 [56.27%]	9.88 [80.05%]	7.70 [87.90%]	5.90 [90.82%]	5.78
9	225	10.88	10.80 [45.03%]	10.80 [63.76%]	10.04 [82.46%]	7.85 [88.37%]	5.89 [90.83%]	5.77
10	250	10.77	10.69 [49.60%]	10.68 [67.29%]	9.92 [83.99%]	7.71 [88.90%]	5.87 [90.83%]	5.75

Table 4.11: Decimal digits of precision of E_{HF} for TIP4P water clusters using the 6-31G** Basis Set, with different cutoff points (λ_{GPU}) where ERIs are evaluated using single precision instead of double precision floating point. D.P stands for a fully double precision ERI calculation, S.P stands for a fully single precision ERI calculation, with cutoff points of various magnitudes in between (The numbers in the square brackets indicate the percentage of single precision ERIs)

all integrals with Schwarz upper bound below λ_{GPU} are calculated using single precision floating point interval arithmetic, while the rest are calculated using the default double precision floating point interval arithmetic.

Results obtained with different values of λ_{GPU} are given in Table 4.11 for water cluster systems using the 6-31G** basis set. Values for λ_{GPU} of 10^{-12}, 10^{-8}, 10^{-4}, 10^{-2}, and 1 are used. The columns labelled D.P and S.P correspond to results calculated using solely double or single precision arithmetic respectively. The values in the square brackets denote the percentage of the unique, non-zero, ERIs that were evaluated using single precision arithmetic.

As expected the number of decimal digits of precision in E_{HF} generally decreases as the problem size increases. One exception is $(H_2O)_3$, where the results obtained using $\lambda_{GPU} = 1 \times 10^{-4}$ and $\lambda_{GPU} = 1 \times 10^{-2}$ are less precise than the equivalent numbers for the larger $(H_2O)_4$ and $(H_2O)_5$ systems. Upon further investigation, we found that this is likely to be due to the composition of the ERIs; $(H_2O)_3$ has more ERIs with magnitudes between 1×10^{-6} and 1×10^{-4} (38%) compared to $(H_2O)_4$ (26%) and $(H_2O)_5$ (20%). Thus, a disproportionately large number of integrals near the threshold are calculated in single precision for the $(H_2O)_3$ case. The same explanation can be given in the case of $(H_2O)_8$ in relation to $(H_2O)_9$ and $(H_2O)_{10}$ for $\lambda_{GPU} = 1 \times 10^{-4}$ and $\lambda_{GPU} = 1 \times 10^{-2}$. This shows that while partitioning between single and double precision calculation based solely on the Schwarz upper bound of the ERI is generally reliable, the specific distribution of integral values can also be a factor for some systems. Since the magnitude of a two electron integral is determined by the distance between each electron, we can

	λ_{GPU}					
D.P	$\mathbf{1 \times 10^{-12}}$	$\mathbf{1 \times 10^{-8}}$	$\mathbf{1 \times 10^{-4}}$	$\mathbf{1 \times 10^{-2}}$	$\mathbf{1 \times 10^{+0}}$	**S.P**
4.65×10^{-8}	4.14×10^{-8}	4.12×10^{-8}	4.78×10^{-8}	4.98×10^{-6}	4.53×10^{-5}	5.97×10^{-5}

Table 4.12: Predicted relative numerical error at $N = 10,000$ for TIP4P water clusters for different λ_{GPU}.

expect denser molecular systems to have higher concentrations of large integrals.

The results also show that numerical errors increase as the cutoff is relaxed and more integrals are calculated in single precision. For the problems considered, using a λ_{GPU} value as large as 10^{-2} still provides sufficient precision to give E_{HF} values that are chemically accurate, while using a value for λ_{GPU} of 1 is clearly inadequate. Alternatively setting a λ_{GPU} threshold of 10^{-8} gives results that are almost as precise as exclusively using double precision.

In Section 4.5.1 we used a second-degree polynomial regression model to predict numerical errors when $N = 10,000$. If we apply the same approach to the data in Table 4.11 we find the projections for $N = 10,000$ given in Table 4.12. The columns show the predicted (relative) numerical errors for each respective λ_{GPU} value. If only single precision is used, the predicted error is of the order of 1×10^{-5}, which is not chemically accurate.

4.8.2 Effect of $F_m(T)$ Interpolation Table Size on the Total Energy

Memory management is one of the most important factors affecting the performance of CPUs, as well as GPUs and other special purpose processors. This can mean managing a complex cache hierarchy, small amounts of user controlled local memory, or a combination of both. This problem is especially acute on GPUs where thousands of threads must contend for fast memory smaller than that on a conventional CPUs. For example the NVIDIA GTX280 only has 16KB of fast local shared memory available to each thread block [114], the synergistic processing elements (SPE) on the STI CellBE each have 256KB of local memory for SPE instructions and data [62], and the SPARC VIIIfx CPU has a user controlled on-chip memory that can be cache, local memory or a combination of both[99].

How to use small fast on-chip memory is therefore an increasingly important question for program developers. In ERI evaluation, the $F_m(T)$ interpolation table is a good candidate for storage in fast memory; use of interpolation gives rise to a significant reduction in floating point operations, but this is only beneficial if the data that comprises the interpolation table can be accessed quickly. So an

issue is how small can the $F_m(T)$ interpolation table be made while still obtaining sufficiently accurate results. For the purpose of this study we will assume an ERI evaluation scheme similar to that employed by Ufimtsev and Martinez where each thread or thread block on a GPU calculates a single primitive integral or contracted integral, with the $F_m(T)$ interpolation table replicated for each thread block in the fast local memory.

The size of the interpolation table depends on i) the interpolation scheme used, ii) the degree of the polynomial interpolation, iii) the largest total angular momentum of any ERI, and iv) the truncation error tolerance. Table 4.13 shows the amount of memory required to represent a third and sixth degree Chebyshev polynomial interpolation table for integrals of type $(ss|ss)$ through to $(gg|gg)$ when using different $F_m(T)$ truncation error tolerances. Third degree polynomials require larger interpolation tables than sixth degree polynomials, but require less operations to evaluate each subsequent $F_m(T)$ values. Thus the choice of whether to use a third or sixth order polynomials involves a trade-off between floating point operation count and memory usage. For GPUs and the CellBE systems, memory size is the major bottleneck, thus sixth degree interpolation appears to be the best choice. The results in Table 4.13 show that to keep truncation error around machine precision requires an interpolation table much larger than that which can be stored in the shared memory on a current GPU system. However, by relaxing the error tolerances, it is possible to fit the interpolation table in fast shared memory. Supposing that 16KB of shared memory is available per thread block, then a maximum table size of 8KB (or less) is appropriate as it allows room for other intermediate data to be stored in shared memory, and allows up to 2 thread blocks to execute concurrently on each SM. If 8KB is the maximum table size allowed, then using sixth-degree Chebyshev polynomials $(ss|ss)$ type integrals can be calculated with errors in the order of 10^{-12}, and $(pp|pp)$, $(dd|dd)$, $(ff|ff)$, and $(gg|gg)$ integrals with errors in the order of 10^{-8}.

The approaches outlined in Section 3.3 can be used to place error bounds on $F_m(T)$ values under different levels of interpolation table granularity; which, can then be propagated in order to assess the overall impact on the precision of E_{HF}. The result of this is shown in Table 4.14 for sixth-degree Chebyshev polynomial interpolation and a range of different $F_m(T)$ thresholds. The results show that the precision of E_{HF} is decreased when the error tolerance is relaxed, and that chemical accuracy is maintained in all cases except when the $F_m(T)$ threshold is 10^{-4}. However, if only single precision arithmetic is used then a threshold slightly less than 10^{-4} will suffice to maintain the 5 to 6 decimal digits of precision predicted by Table 4.11. Interestingly, and in a similar manner to what was observed in Section 4.8.1, when using error tolerances of 10^{-8} and 10^{-4} the

		$e_{F_m(T)}$			
Integral	**Degree**	ϵ	$\mathbf{1 \times 10^{-12}}$	$\mathbf{1 \times 10^{-8}}$	$\mathbf{1 \times 10^{-4}}$
(ss|ss)	3	1134	138	13	1
	6	25	7	2	<1
(pp|pp)	3	1581	193	19	1
	6	35	10	2	<1
(dd|dd)	3	1925	235	23	2
	6	43	12	3	1
(ff|ff)	3	2269	277	27	2
	6	50	15	4	1
(gg|gg)	3	2544	310	31	3
	6	56	17	4	1

Table 4.13: Chebyshev Interpolation Table Sizes (KB) for different integral types and truncation error tolerances.

			$e_{F_m(T)}$			
System	**Basis**	**N**	ϵ	1×10^{-12}	1×10^{-8}	1×10^{-4}
$(H_2O)_4$	6-31G**	100	11.69	11.68	9.05	4.97
Alanine	6-31G**	125	11.41	11.37	8.40	4.22
Serine	6-31G**	140	11.30	11.27	8.39	4.19
Cytosine	6-31G**	145	11.24	11.21	8.31	4.14
$(H_2O)_8$	6-31G**	200	11.00	11.00	8.85	4.74
$(H_2O)_{10}$	6-31G**	250	10.77	10.68	8.88	4.78
	Table Size (KB)		43	12	3	1

Table 4.14: Decimal digits of precision of E_{HF} for different $F_m(T)$ truncation error tolerances (e_{FMT}). The last row indicates the interpolation table sizes associated with each tolerance.

precision for $(H_2O)_8$ is greater than for any of the amino acid cases despite the fact the total number of basis functions (N) in the former is greater than the latter. This is again because there is a higher proportion of large magnitude ERIs for the amino acids than for $(H_2O)_8$ and $(H_2O)_{10}$.

4.8.3 Summary: Numerical Errors in GPU Computation

The results highlight the fact that under certain conditions chemical accuracy can still be achieved even when the Hartree-Fock energy is calculated using relatively unconventional parameters, for example i) by varying the arithmetic precision between single and double precision, and ii) by varying the granularity

of integral interpolation tables. This result gives reasons to suggest that key modifications to existing Hartree-Fock evaluation schemes can be made while maintaining chemical accuracy. This is especially relevant when looking to work around the limitations of specialized processing hardware in regard to double precision performance, and fast memory capacity. On a cautionary note, the results also show that there are other factors, besides system size, that affect accuracy such as the density of the molecular system.

4.9 Errors in Fock Matrix Diagonalization

So far we have investigated numerical errors in the Hartree-Fock total energy calculations under different input and design related conditions. The following section now examines the numerical errors propagated as a result of Fock matrix diagonalization, and its effect on the precision of quantities including i) the molecular orbital energies, the ii) molecular orbital coefficients, and iii) derived electronic properties from population analyses. The experiments are performed using the interval Fock matrix diagonalization approaches described in Section 3.5.

The results are presented as a series of examples.

Example 4.1. (Molecular Orbital Energies of $(H_2O)_2$/3-21G) The following example shows the molecular orbital energies of a TIP4P $(H_2O)_2$ water cluster, calculated using the 3-21G basis set. The energy values and their respective levels of precision (in decimal digits) are summarized as follows:

	Molecular Orbital Energy		Digits of Prec.
0	[-20.47727562780773,	-20.47727562759915]	10.99
1	[-20.37132608218376,	-20.37132608197516]	10.99
2	[-13.74298987314374,	-13.74298987105788]	9.82
3	[-12.79967982520638,	-12.79967982312052]	9.79
4	[-7.285680065575629,	-7.285680063489783]	9.54
5	[-6.571546550694757,	-6.571546548608911]	9.50
6	[-5.811746898493541,	-5.811746896407695]	9.45
7	[-5.269789747013400,	-5.269789744927554]	9.40
8	[-4.809489735153614,	-4.809489733067768]	9.36
9	[-4.316967159087162,	-4.316967157001316]	9.32
10	[0.2313156974985794,	0.2313156977071640]	9.04
11	[0.3318598499019961,	0.3318598501105807]	9.20

12	[0.3332822511724854,	0.3332822513810700]	9.20
13	[0.5139545197717393,	0.5139545199803239]	9.39
14	[1.157505779976690,	1.157505780185275]	9.74
15	[1.233885074427448,	1.233885074636033]	9.77
16	[1.297949165405931,	1.297949165614517]	9.79
17	[1.371356514744519,	1.371356514953105]	9.82
18	[1.764689765950226,	1.764689766158812]	9.93
19	[1.828866800008023,	1.828866800216609]	9.94
20	[1.907811065674676,	1.907811065883262]	9.96
21	[1.922580091566733,	1.922580091775318]	9.96
22	[1.970394794565698,	1.970394794774284]	9.98
23	[2.152045398872705,	2.152045399081290]	10.01
24	[3.050970127688570,	3.050970127897155]	10.17
25	[3.398902681390923,	3.398902681599506]	10.21

This result shows that numerical precision increases with the magnitude of the M.O energy. This is not indicative of error behaviour as such, but more likely due to the application of Rohn's basic bounding method [123] (see Equation 3.33). Since the absolute width of all interval eigenvalues obtained using this method are equal to the spectral radius of the radius matrix, the relative error of the smaller eigenvalues will naturally be larger. In any case, the results show that the M.O energies can be evaluated within 9 to 10 decimal digits of precision. The overall root mean squared error (RMSE) of the all M.O energies combined is equivalent to 9.79 decimal digits of precision. Although this is an acceptable level of precision, it is not as precise as the total energy evaluated for the same system which can be evaluated within 12 decimal digits of precision.

The eigenvectors of the Fock matrix correspond to a new set of molecular orbital coefficients. If the M.O coefficients used to construct the Fock matrix corresponds to the ground state, then, in principle, the new M.O coefficients should be identical to it. However, in practice, the new M.O coefficients will almost always be different due to numerical errors. Interval matrix diagonalization allows rigorous error bounds to be placed on these new M.O coefficients.

The M.O coefficients can be used to perform population analyses, which provides a means of estimating the partial atomic charges of atoms within the system. In *Mulliken's population analysis* [148], the approximate atomic charge of an atom A is written as

$$q_A = Z_A - \sum_{\mu \in A} (PS)_{\mu\mu} \tag{4.9}$$

where Z_A is the charge of the nucleus of A, P is the density matrix, and S is the overlap matrix. Note that the summation is carried out only with respect to the basis functions that are centered on A.

Evaluating error bounds on the M.O coefficients allows us to represent the density matrix P in terms of an interval matrix, which then allows error bounds on each partial atomic charge, q_A, to be computed.

Example 4.2. (Partial Atomic Charges of $(H_2O)_2$/3-21G) The following example shows the Mulliken's population analysis of a TIP4P $(H_2O)_2$ water cluster calculated using the 3-21G basis set. The error bounds of each partial atomic charge are as follows:

		Partial Atomic Charges		Digits of Prec.
1^{st}	O	[-.8008763680548691,	-.8008755571308663]	5.99
	H	[0.3406691782077763,	0.3406693139197349]	6.40
	H	[0.3966614490214185,	0.3966616441813792]	6.31
2^{nd}	O	[-.7174418399681320,	-.7174414741909375]	6.29
	H	[0.3904933964642431,	0.3904934379696717]	6.97
	H	[0.3904933905989220,	0.3904934289816099]	7.01

The results show that partial atomic charges can be evaluated within 6 to 7 decimal digits of precision. The RMSE of all the partial atomic charges combined is equivalent to 6.51 decimal digits of precision, which is not as precise as the molecular orbital energies, or the total energy of $(H_2O)_2$ / 3-21G. This is likely the consequence of the additional computational steps required to first find the interval enclosure of the eigenvalues and the eigenvectors, and then evaluate Equation 4.9.

The numerical precision of the i) total energy, ii) molecular orbital energies, iii) molecular orbital coefficients, and iv) partial atomic charges for 3-21G water clusters of different sizes are shown in Table 4.15. The results are reported in terms of decimal digits of precision obtained from the RMSE of each quantity.

As expected, the results show that the numerical precision of each quantity generally decreases as the problem size increases. The total energies can be evaluated more precisely than any of the other quantities, by several orders of magnitude. This is likely to be a reflection of the difficulty in obtaining tight bounds on the eigenvalues and eigenvectors of an interval matrix, and not necessarily the actual numerical errors itself. However, when the interval results are within the accuracy required, we do have a strong assurance of numerical accuracy on each of these quantities.

		3-21G			
$(H_2O)_n$	**N**	Total Energy	M.O Energies	M.O Coeff	Atomic Charges
2	26	12.63	9.79	8.03	6.51
3	39	12.30	9.18	7.22	6.02
4	52	12.08	8.63	7.12	5.70
5	65	11.87	8.58	6.44	5.27
6	78	11.70	8.37	6.21	5.20
7	91	11.56	8.11	5.92	4.92
8	104	11.41	7.23	5.55	3.99
9	117	11.28	7.82	5.65	4.49
10	130	11.17	7.60	5.24	4.15

Table 4.15: Root mean squared error of the i) total energy, ii) molecular orbital energies, iii) molecular orbital coefficients, and iv) partial atomic charges, expressed in terms of decimal digits of precision.

4.10 Related Work

Takashima et. al. [149] investigated the numerical accuracy of several different schemes for calculating $F_m(T)$. Their work focused in particular on the effect of rounding errors, which were measured by computing the mean squared difference between the set of $F_m(T)$ values computed using double precision (52-bit) arithmetic, and the set of $F_m(T)$ values computed using extended double precision (64-bit) arithmetic. The latter is assumed to be the correct result.

Takashima et. al. [150] also studied the numerical accuracy of large scale Hartree-Fock calculations. Numerical errors in the total energy were measured by comparing total energy values computed with, and without, error perturbed (or contaminated) input data. The perturbations were introduced by uniformly replacing the last bit of the mantissa of the input data (two electron integrals and density matrix elements). Systems with up to $N = 427$ basis functions were evaluated. The results show a general increase in numerical errors as the number of basis functions increases. A linear regression generated from existing data was applied to predict errors for large scale systems of up to 10,000 basis functions. It was then demonstrated that the accuracy can be improved significantly by using partial summation in the Fock matrix construction step.

Takashima also investigated numerical errors introduced in the Fock matrix diagonalization step. The authors did not consider actual Fock matrices, but, instead, two types of surrogate matrices. The first is based on a class of matrices developed by Frank [33], for which analytical eigenvalues and eigenvectors can be

evaluated. The second are a set of randomly generated real symmetric matrices. Computational results were given for eigenvalues and eigenvectors of matrices as large as 1024×1024.

While the work by Takashima et. al. has many similarities with the work presented in this chapter, it is difficult to compare the two sets of results. The approach adopted by Takashima is in essence a one variable at a time (OAT) approach which involves fixing uncertain variables or parameters to an assumed (perturbed) value. It does not consider alternate assumptions, for example, as to how the input data should be perturbed. While an OAT approach can be implemented using existing software, and allows large models to be considered, it is likely to produce results that *underestimate* the actual numerical errors. The interval analysis approach used in this chapter, on the other hand, is able to encapsulate the entire range of alternate assumptions, and produce results that (strictly) *overestimate* the actual numerical errors.

Recent interest shown in performing electronic structure calculations using specialized processor hardware including FPGAs, CellBE, GPUs have naturally led to investigations of numerical accuracy, especially in situations when different levels of arithmetic precision are used.

Ramdas et. al. [120] proposed the use of interval arithmetic as an *off-line design space tool* to experiment with different arithmetic unit configurations. In particular, they were interested in whether a mix of different arithmetic precision level could be used to perform Hartree-Fock calculations on an FPGA, while maintaining an acceptable degree of accuracy. Their investigation was, however, confined to evaluating the interval width of G-matrix elements (see Equation 2.15) for a single H_2O molecule using the 6-31G** basis set.

Yasuda [162] presents an algorithm based on the Rys quadrature method for evaluating two-electron integrals on graphics processing units. Considerable attention in this paper was devoted to investigating numerical errors arising form GPU computation. To minimize the memory footprint, a theoretical analysis was performed to determine the minimum level of arithmetic precision required to represent the roots and weights of the Rys polynomial required to ensure an ERI accuracy of 10^{-7} (absolute error).

To strike an effective balance between performance and accuracy, Yasuda proposed a threshold value λ_{GPU}, where ERIs with magnitude below the threshold are computed in 32-bit single precision on the GPU, while the remaining larger ERIs (where accuracy is more important) are calculated in 64-bit double precision on the CPU. This partitioning scheme was used as a basis to perform, mixed GPU and CPU, Self-consistent field (SCF) calculations.

Yasuda also investigated numerical errors in two-electron integrals and total energies for different values of λ_{GPU}. The experiments focused on Taxol and Valinomycin using the 3-21G and 6-31G basis sets. Numerical errors were measured by comparing the computed result to a reference value computed entirely in double precision. The results demonstrate that the error of the total energy decreases to an acceptable threshold, as λ_{GPU} decreases. On the other hand, the results also show that GPU-only calculations, performed entirely in single precision, are not sufficiently accurate for production runs (based on the millihartree threshold or 300K). However, it was also found that GPU-only calculations did not affect the stability of SCF convergence, and may still be useful in providing accurate initial guesses. The findings by Yasuda were, as a whole, consistent with the results provided in Section 4.8.1.

Ufimtsev and Martinez [152, 153, 154] demonstrated the application of graphics processors to large scale Hartree-Fock point energy calculation [152, 153], and geometry optimization [154]. While the authors chose to focus primarily on calculations in 32-bit single precision, they were also mindful of errors that might occur as a consequence. The most relevant points are discussed in [153].

Ufimtsev and Martinez [153] used double precision results obtained from the GAMESS quantum chemistry package [136] as reference points from which to evaluate the accuracy of their proposed GPU implementation. The accuracy of two different GPU implementations were compared, one computed entirely in 32-bit single precision, and the other using a mix of 32-bit single precision and 64-bit double precision (the latter is possible with the NVIDIA GTX280). The ERIs in both implementations were computed in single precision; however, in the mixed precision case, the accumulation of ERIs to form various matrix elements are performed in double precision to reduce errors. Ufimtsev considered systems with as many as 453 atoms and 2131 basis functions. Their error analysis demonstrated that chemical accuracy at the millihartree level, relative to GAMESS, can be maintained in all cases using the mixed precision scheme. However, it is unclear whether their analysis was affected by the SCF convergence threshold used by GAMESS and their GPU code, given that the margin of error reported could be of a similar magnitude. Perhaps due to this, the results do not indicate a consistent loss of precision due to increasing problem size, which contradicts what was observed in this chapter and in Takashima [150]. Finally, the authors observed that there was no SCF convergence instability as a result of using single precision, which is consistent with a similar observation by Yasuda.

The methodology employed to measure numerical errors by Yasuda [162], and Ufimtsev and Martinez [152, 153, 154] are similar to the OAT approach employed by Takashima et. al. [150, 149]. They made an assumption that there exists a

reference value from which to compare the accuracy of computed results with. They do not consider, however, how accurate the reference value is to the exact result, or the influence of other sources of error besides the shift from double precision to single precision computation.

In a recent publication, Luehr et. al. [93] presented a technique for dynamically setting λ_{GPU} in Hartree-Fock and density functional Self-Consistent Field calculations. The motivation for varying λ_{GPU} came from the observation that relatively large errors can be tolerated in the earlier SCF iterations without hampering convergence. The maximum element of the DIIS error vector [119] was used as the metric for the tolerance associated with each iteration. The value of λ_{GPU} required to satisfy a particular error tolerance is predicted based on experimental data.

Knizia et. al. [72] introduced a generalized method for determining the numerical properties of electronic structure algorithms. This method is based on the statistical sampling of output variability through the addition of random noise. To implement this method, a script is used to instrument the original code so that the rounding modes of selected floating point operations are varied in a random fashion at runtime. The instrumented code is then executed multiple times so that the numerical scattering due to the noise can be analyzed. This approach is both simple and robust, and can facilitate the study of a wide range of numerical issues. However, a major limitation is the running time, which was reported to be 5 to 50 times slower than the original code due to the number of samples required.

4.11 Conclusion and Future Work

The *Interval Hartree-Fock* program was used as an error analysis tool to explore the effect of numerical error in Hartree-Fock computations.

The chapter begins by discussing the application of interval analysis to error analysis. The key observation is that unlike traditional sampling based approaches, interval analysis can be used to study i) models with large number of dimensions, and ii) sources of error that cannot be described by a probability distribution.

The numerical errors in evaluating the reduced incomplete gamma function $F_m(T)$ was investigated. The results showed that, in general, $F_m(T)$ can be evaluated to within 14 to 15 decimal digits of precision. The most accurate results were obtained by using compensated summation or cascaded accumulated summation to evaluate Shavitt's series. On the other hand, using third-degree

Chebyshev interpolation together with Horner's method yielded inaccurate results due to cancellation error.

The accumulation of numerical errors due to increasing basis set and system size was studied in water clusters using the 3-21G, 6-31G**, cc-pVDZ, 6-31++G, and 6-311G (2df,2pd) basis sets. We showed that it is possible to accurately describe the growth in numerical error using a second degree polynomial regression model. It was also shown that numerical error is reduced significantly when accurate summation techniques are applied to compute the total energy.

Numerical errors in the total energy due to different two-electron integral pre-screening thresholds were also investigated. The results confirmed that chemical accuracy can still be maintained for certain thresholds.

The behaviour of numerical errors in computation performed on graphics processing units (GPU) were investigation. This investigation focused on the repercussions of attempts to work around the limitation of GPUs in relation to double precision performance, and fast memory capacity. In particular, the questions over the impact of i) evaluating two-electron integrals using a combination of single and double precision arithmetic, and ii) varying the granularity of the $F_m(T)$ interpolation table. The results highlighted the fact that chemical accuracy can be achieved in both scenarios provided that the appropriate tolerances are applied.

Finally, we investigated the numerical errors propagated by Fock matrix diagonalization, and its effect on quantities including the molecular orbital energies, the molecular orbital coefficients, and partial atomic charges.

The main drawback of using interval analysis for numerical accuracy studies is that it provides a worst-case error analysis. The fact that interval results get rounded out to the nearest larger machine representable interval and the dependency problem, means that these results are almost always overly pessimistic. That said, when the interval result is within the accuracy required we have a rigorous statement of numerical accuracy.

The current results also rely on a conventional Self-Consistent-Field (SCF) calculation to find the ground state molecular orbitals of a fixed atomic geometry. From the results of this calculation, the interval total energy and associated quantities are then calculated. This work does not consider how accurate a solution these coefficients are to the Hartree-Fock equations, or indeed whether the solution represents the correct result. This shortcoming in our analysis is due to the difficulty in ensuring the correctness of solutions obtained using iterative methods based on local optimization. The essence of which will be addressed in the following chapters.

Part

III

Deterministic Global Optimization Approaches

The following chapters present our efforts *to compute solutions for the Hartree-Fock equations that are guaranteed to be within a specified margin of error* through the application of deterministic global optimization techniques.

Chapter 5 provides the background to deterministic global optimization, and outlines methods for handling quantum chemistry specific mathematical expressions within its framework. A program is presented for evaluating the Hartree-Fock equations using deterministic global optimization.

Chapter 6 uses this program to provide proof of concept results in i) point energy calculations, ii) geometry optimization, iii) basis set optimization, and iv) excited state calculations. A performance analysis is also presented.

Chapter

5

Development of a Deterministic Global Optimization Approach for Hartree-Fock Theory

Contents

Many problems in electronic structure theory can be posed as one of finding the global minimum of an energy expression. These problems usually fall under a more general class of problems known as *Non-Linear Programming* (NLP)

$$
\begin{array}{lllll}
P: & z = \min & f(x) & & \\
& \text{s.t.} & g_j(x) \leq 0 & \forall j \in \{1,2,3,..,q\} & \\
& & h_k(x) = 0 & \forall k \in \{1,2,3,..,r\} & \\
& & x_i^l \leq x_i \leq x_i^u & \forall i \in \{1,2,3,..,n\} &
\end{array}
\tag{5.1}
$$

where $x \in \mathbb{R}^n$, $f : \mathbb{R}^n \to \mathbb{R}$, $g_j : \mathbb{R}^n \to \mathbb{R}$, $h_k : \mathbb{R}^n \to \mathbb{R}$, $x_i \in \mathbb{R}$, $x_i^l \in \mathbb{R}$, $x_i^u \in \mathbb{R}$; and $f(x)$, $g_j(x) \leq 0$ and $h_k(x) = 0$ may be non-linear functions.

Using the variational principle, the *Restricted* Hartree-Fock equations (RHF) for closed shell systems consisting of N_{elec} (even numbered) electrons and N_{atom} atoms can be expressed as a continuous, non-convex NLP problem of the form [79]

$$
\begin{array}{lllll}
RHF: & E_{HF} = & \min 2[& \sum_{\mu=1}^{N_{elec}/2}\{(\phi_\mu| - \frac{1}{2}\triangle^2|\phi_\mu) - \sum_{A=1}^{N_{atom}}(\phi_\mu|\frac{Z_A}{r_{iA}}|\phi_\mu)\} & \\
& & & \sum_{\mu=1}^{N_{elec}/2}\sum_{\upsilon=1}^{N_{elec}/2}\{(\phi_\mu\phi_\mu|\phi_\upsilon\phi_\upsilon) - \frac{1}{2}(\phi_\mu\phi_\upsilon|\phi_\mu\phi_\upsilon)\}] + V_{NN} & \\
& s.t. & & (\phi_\mu|\phi_\upsilon) = \delta_{\mu\upsilon}, \quad \forall \mu, \upsilon = \{1,2,3,..,N_{elec}/2\} &
\end{array}
\tag{5.2}
$$

where the total (Restricted) Hartree-Fock energy (E_{HF}) is minimized, subject to the orthonormality of the set of molecular orbitals $\{\phi_\mu\}_{\mu=1}^{N_{elec}/2}$. The exact RHF formulation depends on the nature of the computation being performed. For example, in a *point energy calculation*, the ground state wavefunction is usually determined by minimizing E_{HF} with respect to the form of the molecular orbitals, while keeping the location of the nuclei and the parameters defining the basis set fixed. On the other hand, in a *geometry optimization*, the equilibrium geometry is usually obtained by minimizing E_{HF} with respect to the molecular orbitals and the location of the nuclei, while keeping the parameters defining the basis set fixed. Although both of these problems are non-convex, the traditional approach has been to apply iterative numerical methods equivalent to local optimization. For example, the Roothaan Self-Consistent Field (SCF) method is, in essence, a fixed point iteration scheme used to find the zero gradient of the Lagrangian dual of RHF [148]. The success of these approaches depend, in a probabilistic sense, on finding a good starting point from which to begin the iterations. This is not guaranteed to succeed, which means that there is a degree of uncertainty associated with almost all results computed in this way.

The global optimization of non-convex problems is NP-Hard in the general instance [130]. While most local optimization methods are based on greedy algorithms that tend to converge towards the nearest local minima, *global optimization* methods make a more concerted effort to find the global minima. Global optimization approaches can be classified under two broad categories; *Stochastic Global Optimization* [50] and *Deterministic Global Optimization* [50, 26, 66].

Stochastic global optimization usually involves generating random starting points for local optimization in the hope of generating one that is sufficiently close to the global minimizer. The best feasible solution found in this process is regardless declared to be the global minimizer. Therefore, the optimality of stochastic global optimization can only be guaranteed in a stochastic sense.

In this thesis we focus on solving the Hartree-Fock equations using the more rigorous deterministic global optimization approach. These methods are *deterministic* in the sense that they can provide strong mathematical guarantees that the solution is ϵ-optimal (by Definition 2.29). Although this often necessitates some form of worst-case exponential time search, many useful applications in science and engineering have started to emerge as this area begins to mature, and as more advanced computing hardware is deployed [27]. Furthermore, the ability to produce mathematically rigorous solutions is significant to a broad range of problems.

This chapter outlines the development of a deterministic global optimization solver for problems in Hartree-Fock theory. This solver is implemented as an extension to an existing open source general-purpose deterministic MINLP solver, COUENNE (*Convex Over and Under ENvelopes for Nonlinear Estimation*). The main contributions of this work concern the development of algorithms for handling quantum chemistry specific mathematical expressions - such as the one and two-electron integrals - within a deterministic optimization framework. The applications targeted using this approach include point energy calculation, geometry optimization, basis set optimization, and excited state calculation at the Hartree-Fock level. The approaches developed in this chapter can be generalized further to include other electronic structure methods. It can also be used to provide rigorous *computer assisted proofs* for theorems that involve one and two-electron integrals, which may have significant applications to algorithm development within this domain.

This chapter is organized as follows. Section 5.1 provides a brief introduction to stochastic global optimization. Section 5.2 gives the background to deterministic global optimization. Section 5.3 discusses the mathematical programming formulation of the Hartree-Fock equations. Section 5.4 describes methods for solving the Hartree-Fock equations using deterministic global optimization. Section 5.5 discusses related work. The chapter is concluded with Section 5.6.

5.1 Stochastic Global Optimization

Stochastic global optimization methods usually involve the generation of random variable or functional starting points from which to perform local optimization. The hope is to generate eventually one that leads to convergence towards the global minimum, and not merely a local minimum or a saddle point. Whether or not this is actually the case, the best feasible solution found is declared to be the global minimizer.

The success of this approach depends on the strategy applied to select the starting point. Most of these are based on i) randomly selecting points that correspond to a particular sampling distribution, ii) randomly augmenting the formulation of the original problem, or iii) selecting a random starting point that is outside a known or suspected region of attraction. Examples of these approaches include:

- Sobol Sampling [75]
- Stochastic Tunnelling [160]
- Basin Hopping [158]
- Variable Neighbourhood Search [106]
- Simulated Annealing [71]
- Particle Swarm Algorithms [70]
- Genetic Algorithms [48]

Even though the optimality of solutions produced in this way can only be guaranteed in a stochastic sense, stochastic optimization has been of great practical significance, especially in solving small to medium scale global optimization problems [27, 163].

5.2 Deterministic Global Optimization

Deterministic global optimization techniques are able to guarantee global optimality to within a small threshold ϵ. This property is referred to as ϵ-optimality [26]. Supposing $\underline{z}$ and $\overline{z}$ are the lower and upper bounds of the global minimum, ϵ-optimality (by Definition 2.29) requires

$$\overline{z} - \underline{z} < \epsilon. \tag{5.3}$$

To determine either of these bounds requires a high degree of global knowledge that inevitably demands some form of rigorous search over the entire feasible domain. For practical purposes we can think of the lower bound $\underline{z}$ as *the best possible solution*, and the upper bound $\overline{z}$ as *the best feasible solution found so far*. Both quantities can be continually updated via a systematic search of the problem domain. Convergence is this case is referred to as ϵ-*convergence*.

The following sections provide an overview of the necessary components of a deterministic global optimization solver. The optimization problem being solved is assumed to be P as in Equation 5.1. Furthermore, all references to feasibility or infeasibility refer, respectively, to ϵ-feasibility and ϵ-infeasibility (by Definition 2.30), unless stated otherwise.

5.2.1 Spatial Branch and Bound Method

The *spatial branch and bound* method (sBB) provides one of the most effective algorithmic framework for obtaining ϵ-optimality.

The *branching* step involves the recursive decomposition of the original variable domain, referred to as the *root node* $N_0 \in I(\mathbb{R}^n)$, into a hierarchy of *descendent* nodes $N_k \in N_0, k \in \{1, 2, 3, ..\}$ (see Definition 5.1). These nodes are organized as a binary tree, known as *search tree*, for which N_0 is the root.

Definition 5.1. (Nodes) The root node N_0 with respect to P (Equation 5.1) is defined as

$$N_0 = \{x \mid x_i^l \leq x_i \leq x_i^u,\ \forall i \in \{1, 2, 3, .., n\}\} \tag{5.4}$$

or equivalently in terms of interval vectors as

$$N_0 = \bigcup_{i=1}^{n} \{[x_i^l, x_i^u] \times e_i\} \tag{5.5}$$

where e_i denotes the i^{th} column of the identity matrix $I \in \mathbb{R}^{n \times n}$. The multiplication between the interval $[x_i^l, x_i^u]$ and e_i results in an interval vector that is zero everwhere except for its i^{th} element which is equal to $[x_i^l, x_i^u]$. The set of descendent nodes $N_k, k \in \{1, 2, 3, ..\}$ can be described as non-overlapping vector subspaces of N_0.

The *bounding* step involves finding the lower and upper bound of P relative to a node. Here we denote P_k as the original problem P constrained within the subspace spanned by N_k. Its lower bound ($\underline{z_k} \in \mathbb{R}$) is usually found by solving a convex relaxation of P_k denoted as LP_k, while its upper bound ($\overline{z_k} \in \mathbb{R}$) is found by solving P_k. For this purpose, local minimization techniques can be applied to solve both LP_k and P_k. The specific techniques that are chosen depends on characteristics of each respective problem. Several interesting scenarios are encountered specifically after the bounding step:

- If P_k is infeasible or if $\underline{z_k}$ is greater than $\overline{z}$, then the global minima cannot possibly exist in N_k, thus N_k can be *fathomed* (removed from the tree).
- If $\underline{z_k}$ is the smallest value in the set of lower bounds associated with the nodes in the tree, then it can be used to update $\underline{z}$.
- If $\overline{z_k}$ is smaller than $\overline{z}$, and corresponds to a feasible solution, then it can be used to update $\overline{z}$. The corresponding solution is also stored.
- If $\underline{z_k}$ corresponds to a feasible solution to the original problem P_k (in addition to its relaxation LP_k), then $\underline{z_k}$ is the best possible lower bounding solution of P_k. Thus, N_k can be fathomed since no further improvement can be obtained through branching.
- Else, if the node is not fathomed by this point, then N_k is branched to produce sub-nodes N_k^-, and N_k^+ ($N_k = N_k^- \cup N_k^+$), which are both appended to the search tree.

A more detailed illustration of the spatial branch and bound method is given in Figure 5.1. The labels are explained as follows:

- **(Initialize)** The search tree L is initialized with only the root node N_0, the upper and lower bounds $\underline{z}$, and $\overline{z}$ are set to $-\infty$ and ∞, respectively.
- **(Select Node)** removes a node N_k from L for evaluation. The node N_k with the smallest $\underline{z_k}$ value of all the nodes in L is usually selected. Hence L is usually implemented as a minimum priority queue with $\underline{z_k}$ as the priority value. This gives rise to the sub-problem P_k associated with N_k.
- **(Bounds Tightening)** tightens the Euclidean subspace spanned by N_k using both feasibility and optimality based arguments. See Section 5.2.3 for more detail.
- **(Generate Relaxation)** generates the convex relaxation of P_k denoted as LP_k. See Section 5.2.2 for more detail.

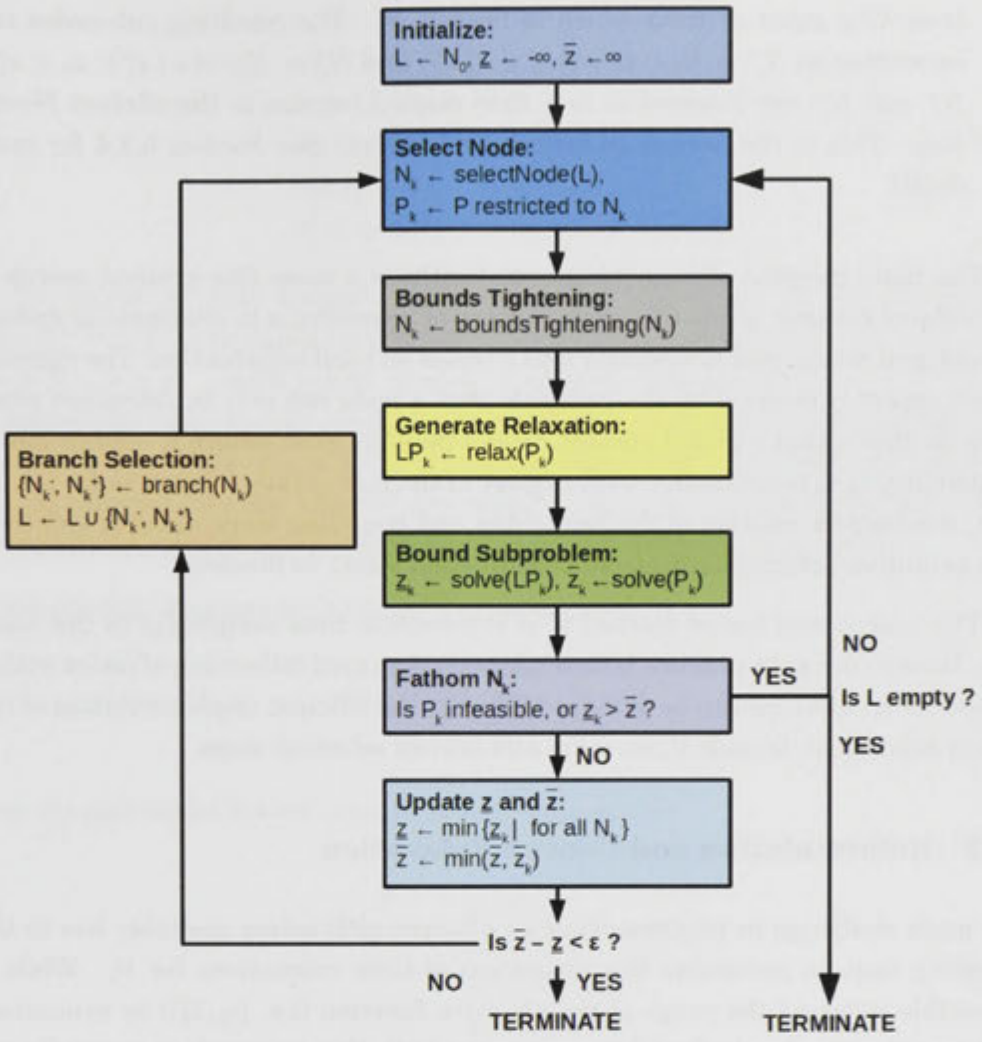


Figure 5.1: A minimal implementation of the sBB algorithm.

- **(Bound Sub-problem)** calculates $\underline{z_k}$ and $\overline{z_k}$, by locally solving LP_k and P_k, respectively. This step is the *bound* in branch and bound.
- **(Fathom)** applies the *fathoming* step discussed above. If N_k is fathomed and L is not empty then return to the **(Select Node)** step. Else if L is empty then **TERMINATE** sBB.
- **(Update $\underline{z}$ and $\overline{z}$)** updates $\underline{z}$ and $\overline{z}$ using the values of $\underline{z_k}$ and $\overline{z_k}$ calculated during the bounding step. If $\underline{z_k}$ or $\overline{z_k}$ is feasible and less than $\overline{z}$ then a new best solution is found and stored as the incumbent.
- **(Convergence)** if $\overline{z} - \underline{z} \leq \epsilon$ therefore ϵ-optimality is satisfied then **TERMINATE** sBB, else proceed to the branch selection step.
- **(Branch Selection)** selects the *branching variable* $x_i \in [x_i^l, x_i^u]$ and the

branching point x_i^b from which to branch N_k. The resulting sub-nodes can be written as $N_k^- = N_k \cap \{x \mid x_i^l \leq x_i \leq x_i^b\}$ and $N_k^+ = N_k \cap \{x \mid x_i^b \leq x_i \leq x_i^u\}$. N_k^- and N_k^+ are inserted in to L then control returns to the **(Select Node)** step. This is the *branch* in branch and bound. See Section 5.2.4 for more detail.

The main purpose of branching is to facilitate a more fine grained search of the problem domain, while the main purpose of bounding is to eliminate or *fathom* non-optimal nodes, and to update $\overline{z}$ and $\underline{z}$ based on local information. The rigorous search aspect is realised by the principle that a node can only be fathomed when it can be shown that a global minima cannot possibly exist within it - either due to infeasibility or sub-optimality with respect to the best solution found so far ($\overline{z}$). As such, the implementation of the branching and bounding steps alone is sufficient for a primitive deterministic global optimization solver to function.

The branch and bound method is of exponential time complexity in the worst case. Its usefulness in practice is dependent on the rapid fathoming of nodes within the search tree; which can be achieved through the efficient implementation of the convex relaxation, bounds tightening, and branch selection steps.

5.2.2 Reformulation and Convex Relaxation

The main challenge in implementing an efficient sBB solver arguably lies in the bounding step, in particular the generation of tight relaxations for P_k. While it is possible to bound the range of the objective function (i.e. $[\underline{z_k}, \overline{z_k}]$) by evaluating its interval extension (referred to as a *range test*), this approach is generally not effective at producing tight bounds due to the tendency for interval analysis to greatly overestimate the range and the fact that the constraints are not taken into account. The most promising approaches so far are based on evaluating the convex relaxation (Definition 2.42) of the primal problem. Convex relaxations are usually simple to generate, and can be solved reliably using local optimization (Theorem 2.40).

The first step in generating a convex relaxation often involves reformulating P into a *standard form* (denoted as $\tilde{P}$). There are a number of different standard forms available. The one considered in this chapter is equivalent to the *Smith Standard Form* [142]. It involves reformulating P into a problem consisting of i) a linear objective function $w(x)$, ii) a set of non-linear equality constraints $x_i = \zeta_i(x)$, iii) a set of linear inequality constraints $v_j(x) \leq 0$, and iv) a set of linear equality constraints $w_k(x) = 0$ given as

$$\begin{array}{lll} \tilde{P}: \quad z = \min & w(x) & \\ \text{s.t.} & x_i = \zeta_i(x) & \forall i \in \{n+1, n+2, n+3, .., n+p\} \\ & v_j(x) \leq 0 & \forall j \in \{1, 2, 3, .., q\} \\ & w_k(x) = 0 & \forall k \in \{1, 2, 3, .., r\} \\ & x_i^l \leq x_i \leq x_i^u & \forall i \in \{1, 2, 3, .., n+p\}. \end{array} \tag{5.6}$$

To arrive at this format, each *non-linear expression* in P, denoted as $\zeta_i(x), \mathbb{R}^{n+p} \to \mathbb{R}$, is replaced recursively by an *auxiliary variable* $x_i, \forall i \in \{n+1, n+2, n+3, .., n+p\}$ - with p new nonlinear equality constraints $x_i = \zeta_i(x)$ added to reflect the relationship between the auxiliary variables and the expressions they replaced. While this is a *lifting* reformulation that introduces new variables and constraints into the original problem, it results in a problem format that is substantially simpler to work with from an implementation perspective.

Example 5.2. Suppose we have the following problem

$$\begin{array}{ll} P: & \min \exp(x_1^2) + x_1 \\ & 0 \leq x_1 \leq 0.5 \end{array} \tag{5.7}$$

then the equivalent Smith's standard form is given by

$$\begin{array}{ll} \tilde{P}: & \min x_4 \\ & x_2 = x_1^2 \\ & x_3 = \exp(x_2) \\ & x_4 = x_1 + x_3 \\ & 0 \leq x_1 \leq 0.5 \end{array} \tag{5.8}$$

where three new auxiliary variables $\{x_2, x_3, x_4\}$ and three new constraints are introduced.

A convex relaxation of $\tilde{P}$ can be generated relatively easily by replacing each set $\{x \mid x_i = \zeta_i(x)\}, \forall i \in \{n+1, n+2, .., n+p\}$ by a convex relaxation that bounds its feasible region. By doing this, we obtain a convex relaxation for $\tilde{P}$ (and therefore P) that consists of a linear objective function and convex constraints (see Definition 2.42 and Definition 2.37).

The convex relaxation scheme applied to each type of non-linear constraint varies between solvers. Currently the most successful approaches are based on constructing a linear relaxation. This is due to the efficiency of Linear Programming (LP) solvers in comparison to general Convex Programming (CP) solvers. Specifically, while a non-linear, convex relaxation may be tighter than a

linear relaxation, the additional time required to solve the former usually offsets any benefits it may have [83, 11].

There are well established procedures [87, 83, 11, 26, 151, 129] for generating LP relaxations for common non-linear expressions types ($\zeta_i(x)$) such as i) bilinear terms (x_1x_2), ii) trilinear terms ($x_1x_2x_3$), iii) quadrilinear terms ($x_1x_2x_3x_4$) iv) quotients (x_1/x_2), v) monomials of even and odd degree (x^{2n}, x^{2n+1}), vi) trigonometric functions ($\sin(x)$, $\cos(x)$, $\tan(x)$), vii) logarithmic functions ($\log(x), \ln(x)$), viii) exponential functions ($\exp(x)$), ix) concave univariate functions, and x) convex univariate functions.

A ubiquitous example is the linear relaxation for bilinear terms $x_3 = \zeta_i(x) = x_1x_2,\ x_i^l < x_i < x_i^u, i = \{1,2,3\}$ devised by McCormick [101]

$$\begin{aligned} x_3 &\geq x_2^l x_1 + x_1^l x_2 - x_2^l x_1^l \\ x_3 &\geq x_2^u x_1 + x_1^u x_2 - x_2^u x_1^u \\ x_3 &\leq x_2^l x_1 + x_1^u x_2 - x_2^l x_1^u \\ x_3 &\leq x_2^u x_1 + x_1^l x_2 - x_2^u x_1^l \end{aligned} \tag{5.9}$$

which defines, in essence, a (linear) tetrahedral envelope around the three-dimensional surface described by the bilinear term [26]. This approach alone is sufficient to generate an LP relaxation for the point energy formulation of RHF.

Another example is the linear relaxation for exponential terms of the form $x_2 = \exp(x_1),\ x_i^l < x_i < x_i^u, i = \{1,2\}$

$$\begin{aligned} x_2 &\geq \exp(x_1^l)(x_1 - x_1^l) + \exp(x_1^l) \\ x_2 &\geq \exp(x_1^m)(x_1 - x_1^m) + \exp(x_1^m) \\ x_2 &\geq \exp(x_1^u)(x_1 - x_1^u) + \exp(x_1^u) \\ x_2 &\leq \frac{\exp(x_1^u) - \exp(x_1^l)}{x_1^u - x_1^l}(x_1 - x_1^l) + \exp(x_1^l) \end{aligned} \tag{5.10}$$

where x_1^m denotes the midpoint between x_1^l and x_1^u. The linear relaxation consists of four linear inequalities. The first three constraints bound $\exp(x_1)$ from below along the tangents of $\exp(x_1)$ at points $x_1 = x_1^l$, $x_1 = x_1^m$, and $x_1 = x_1^u$. The last constraint bounds $exp(x_1)$ from above with the line segment extending from $(x_1^l, \exp(x_1^l))$ to $(x_1^u, \exp(x_1^u))$.

Assuming that $\zeta_i(x)$ is bounded for all $x \in N_k$, then the linear relaxation of $\{x \mid x_i = \zeta_i(x)\}$ can be expressed as a set of linear inequalities $\mathbf{A_i}x \geq b_i$ which forms a polytope around its feasible region. Individual linear inequality constraints in this context are referred to as *linear cuts*.

Given the points discussed above, a linear relaxation of P_k can be expressed formally as

$$\begin{array}{lllll} \mathrm{LP_k}: & \underline{z}_k = \min & w(x) & & \\ & s.t. & \mathbf{A_i}x \geq b_i & \forall i \in \{n+1, n+2, n+3, .., n+p\} & \\ & & v_j(x) \leq 0 & \forall j \in \{1, 2, 3, .., q\} & (5.11) \\ & & w_k(x) = 0 & \forall k \in \{1, 2, 3, .., r\} & \\ & & x_i^l \leq x_i \leq x_i^u & \forall i \in \{1, 2, 3, .., n+p\} & \end{array}$$

where LP_k can be solved using an LP solver such as CPLEX [2], or CLP [31].

While linear relaxations will play a prominent part throughout the course of this chapter, it is also important to be aware of approaches for generating general (non-linear) convex relaxations of P_k. The following section provides a brief introduction to a well known instance: αBB convex relaxation [4, 5].

5.2.2.1 αBB Convex Relaxation

The αBB algorithm [4, 5] is designed to generate convex under-estimators for general, twice-differentiable functions of the form $f : \mathbb{R}^n \rightarrow \mathbb{R}$. The convex under-estimator of $f(x)$ (denoted as $L(x)$) over the domain $[x^l, x^u]$ is defined as

$$L(x) = f(x) + \sum_{i=1}^{n} \alpha_i (x_i^l - x_i)(x_i^u - x_i) \tag{5.12}$$

where $\alpha_i \in \mathbb{R}^+$ are positive scalar parameters. The summation expression $\sum_{i=1}^{n} \alpha_i (x_i^l - x_i)(x_i^u - x_i)$ is a negative, convex function, of which the parameters $\{\alpha_i\}_{i=1}^n$ can be made sufficiently large such that $L(x)$ becomes convex for all $x \in [x^l, x^u]$. Since choosing too large a value for α_i leads to loose underestimators of $f(x)$, the main challenge in αBB is to determine the smallest possible values of α_i that ensures $L(x)$ is a convex relaxation of $f(x)$.

From the theory of convex functions, $L(x)$ is convex if and only if its Hessian matrix $H_L(x) \in \mathbb{R}^{n \times n}$ is positive semi-definite (Theorem 2.35). Moreover, $H_L(x)$ is positive semi-definite if and only if all the eigenvalues of $H_L(x)$ are non-negative [45]. Supposing that $H_f(x) \in \mathbb{R}^{n \times n}$ is the Hessian matrix of $f(x)$, then, from Equation 5.12, $H_L(x)$ can be expressed as

$$H_L(x) = H_f(x) + 2\triangle \tag{5.13}$$

where $\triangle$ is a diagonal $n \times n$ matrix with $\{\alpha_i\}_{i=1}^n$ as its diagonal elements.

Let us consider a simple scenario, suppose that all values of α_i are chosen to be uniformly equal to the value denoted as α. Furthermore, suppose that the eigenvalues of H_f are expressed in increasing order as $\lambda_1(H_f(x)) \leq \lambda_2(H_f(x)) \leq .. \leq \lambda_n(H_f(x))$, then the eigenvalues of H_L can be written as

$$\{\lambda_1(H_f(x)) + 2\alpha, \lambda_2(H_f(x)) + 2\alpha, .., \lambda_n(H_f(x)) + 2\alpha\}. \tag{5.14}$$

For $L(x)$ to be convex, α must be large enough so that all the eigenvalues of H_L become non-negative. Therefore, α can be expressed as

$$\alpha \geq \max\{0, -\frac{1}{2} \min_{x^l \leq x \leq x^u} \lambda_1(H_f(x))\}. \tag{5.15}$$

The problem of finding α becomes one of finding the smallest eigenvalue of $H_f(x)$ for all $x \in [x^l, x^u]$ or its lower bound. This is a similar problem to the interval matrix eigenvalue problem discussed in Section 3.5.2. Several promising $O(N^2)$ and $O(N^3)$-complexity methods have been proposed to solve cases where α_i are uniformly and non-uniformly defined. An extensive outline of these methods, as well as empirical results comparing their effectiveness can be found in [4, 5, 26].

The αBB algorithm was originally designed as a fall-back approach for handling twice-differentiable functions that could not be handled using an existing convex relaxation scheme [4]. Supposing that $f_i(x)$ is such a function, an αBB convex relaxation can be carried out based on the following rules

- If $f_i(x)$ is the objective function of P_k, then replace it by $L_i(x)$.
- If $f_i(x) = 0$ is an equality constraint of P_k, then replace it by two inequality constraints: $L_i(x) \leq 0$ and $-L_i(x) \leq 0$.
- If $f_i(x) \leq 0$ is an inequality constraint of P_k, then replace it by $L_i(x) \leq 0$.

A good feature of the αBB scheme is that, unlike other convex relaxation schemes, it does not introduce any new variables to the original problem regardless of how many non-convex terms each twice-differentiable function encompasses [83].

5.2.3 Bounds Tightening

Bounds tightening is the task of reducing the size of the region spanned by N_k, while preserving the optimal solution. This can help substantially accelerate the

convergence of sBB, by reducing the size of the search space and by improving the tightness of convex relaxations.

In this section three common bounds tightening algorithms are outlined: i) *feasibility based bounds tightening* (FBBT), ii) *aggressive feasibility based bounds tightening* (AFBBT), and iii) *optimality based bounds tightening* (OBBT).

5.2.3.1 Feasibility Based Bounds Tightening

Feasibility based bounds tightening is based on the observation that it is only necessary to search regions that are feasible in P.

Due to the dependencies between the variables of P_k, a change in the bound constraints of one variable, for example due to branching, can potentially trigger changes in the bound constraints of other variables. For instance, it follows from Equation 5.6 that the feasible values of an auxiliary variable x_i is bounded by the range of $\zeta_i(x)$ with respect to $x \in N_k$ as follows

$$\min_{x \in N_k} \zeta_i(x) \leq x_i \leq \max_{x \in N_k} \zeta_i(x). \tag{5.16}$$

Because $\zeta_i(x)$ is a relatively simple function, it is possible to bound its range using interval analysis [66]. If the new range of x_i is narrower than its existing bound constraint, then the bound constraint can be reduced to the new range without losing the optimal solution. The application of Equation 5.16 is referred to as *forward propagation*.

Example 5.3. (Forward Propagation) Supposing we have the constraints

$$\begin{aligned} x_2 &= x_1^2 \\ x_3 &= \exp(x_2) \\ x_4 &= x_1 + x_3 \end{aligned} \tag{5.17}$$

and $x_1 \in [0, 0.5]$, then by forward propagation x_2 must be within the range of $[0^2, 0.5^2] = [0, 0.25]$, x_3 must be within the range of $[\exp(0), \exp(0.25)] = [1, 1.2840]$, and x_4 must be within the range of $[0 + 1, 0.5 + 1.2840] = [1, 1.7840]$.

The relationship between the bound constraints of x_i and x can also be expressed in the reverse order as follows

$$\min\{x \in N_k \mid x_i^l \leq \zeta_i(x) \leq x_i^u\} \leq x \leq \max\{x \in N_k \mid x_i^l \leq \zeta_i(x) \leq x_i^u\} \tag{5.18}$$

where the feasible range of $x \in N_k$ is constrained with respect to $x_i \in [x_i^l, x_i^u]$. The application of this relation is referred to as *backward propagation*. For example suppose we have the constraint $x_3 = \exp(x_2)$ and $x_3 \in [1, 3]$ then by backward propagation, x_2 must be within the range $[ln(1), ln(3)] = [0, 1.0986]$.

Feasibility based bounds tightening (FBBT) is performed by applying forward and backward propagation in alternating order across every constraint in P_k [11]. This process is repeated until no further significant bounds tightening can be obtained, or until a specified number of operations have been performed [11]. FBBT is a relatively inexpensive procedure that can be efficiently applied to every sub-problem considered [11]. FBBT is similar to interval constraint propagation techniques [66, 104].

While performing FBBT, it could be discovered that P_k is *infeasible* if forward or backward propagation yields an *irregular bound*: $x_i \in [x_i^l, x_i^u]$ where $x_i^l > x_i^u$. Furthermore, it could also be discovered that P_k is *non-optimal* if the bounds tightening leads to an objective function whose value exceeds $\overline{z}$. In both of these cases P_k can be fathomed for reasons of infeasibility and non-optimality, respectively.

5.2.3.2 Aggressive Feasibility Based Bounds Tightening

Aggressive feasibility based bounds tightening (AFBBT) is derived from the heuristic assumption that regions distant from a local minimizer are more likely to be infeasible or non-optimal. Supposing that $\hat{x}$ is a local minimizer of P_k, then AFBBT is carried out by applying FBBT to the subspace of N_k that excludes $\hat{x}$.

AFBBT is usually applied on a *per-variable basis*. That is, for each variable $x_i \in [x_i^l, x_i^u]$, FBBT is firstly performed on P restricted to the box $N_k \cap \{x \mid x_i^l \leq x_i \leq \hat{x}_i - \delta\}$ where $\delta > 0$ is a suitable margin. If FBBT finds that the region is infeasible or non-optimal, then $[\hat{x}_i - \delta, x_i^u]$ is a valid tightening for x_i. FBBT is then performed on P restricted to the box $N_k \cap \{x \mid \hat{x}_i + \delta \leq x_i \leq x_i^u\}$. If FBBT finds that the region is infeasible or non-optimal, then $[x_i^l, \hat{x}_i + \delta]$ is a valid tightening for x_i. Note also that AFBBT is equivalent to the *probing* techniques used in MILP and MINLP [11, 129, 151].

AFBBT is considerably more time consuming than FBBT since it requires the computation of a local minimum for P_k, followed multiple executions of FBBT. On the other hand, it is generally more effective than FBBT. To strike a balance between computation time, and bounds tightening, AFBBT is often applied only to nodes above a certain depth (γ^{AFBBT}) in the search tree.

5.2.3.3 Optimality Based Bounds Tightening

Optimality based bounds tightening (OBBT) [142, 83, 11] uses the convex relaxation LP_k (as opposed to P_k) to tighten bounds. By the definition of a relaxation, the feasible set of P_k is also feasible in LP_k, hence the set of variable values that are infeasible in LP_k must also be infeasible in P_k. Thus, a valid tightening for each variable x_i with respect to LP_k (and by implication P_k) can be derived from minimizing and maximizing LP_k using x_i as the objective function,

$$\min\{x_i \in x : \forall x \text{ feasible in } LP_k\} \leq x_i \leq \max\{x_i \in x : \forall x \text{ feasible in } LP_k\}. \quad (5.19)$$

Unlike P_k, it is relatively straightforward to solve problems involving LP_k since it is convex. Although OBBT is generally more effective than FBBT, it is expensive because it involves solving two LP problems for each variable. Like AFBBT it is often only performed on nodes above a specific depth ($< \gamma^{OBBT}$) in the search tree.

5.2.4 Branch Selection

Branch selection refers to the selection of the *branching variable* $x_i \in [x_i^l, x_i^u]$ and the selection of the *branching point* $x_i^b \in [x_i^l, x_i^u]$ from which to partition N_k. In a typical sBB implementation, if the lower bound of P_k is smaller than $\underline{z}$, and if the solution of LP_k is infeasible in P_k then N_k is branched into two *sub-nodes*: $N_k^- = N_k \cap \{x \mid x_i^l \leq x_i \leq x_i^b\}$ and $N_k^+ = N_k \cap \{x \mid x_i^b \leq x_i \leq x_i^u\}$.

An ideal branching algorithm is one that minimizes the number of nodes required for sBB to converge, without itself incurring too high an overhead. As it is difficult to directly predict the sequence of branches that leads towards convergence, let alone find the optimal path, Belotti [11] suggests three distinct *surrogate criteria* that can be used to guide the design of branching algorithms. These criteria include

1. to improve the lower bound of the two resulting sub-problems, or
2. to create sub-problems of similar difficulty, in order to keep the search tree balanced, or
3. to eliminate as large a region as possible from the search space

In practice, it is difficult to satisfy all three criteria at the same time, indeed some may even contradict each other in certain cases. As such, Belotti argues

that it is necessary to use empirical testing to evaluate the performance of each branch selection algorithm [11]. The following section briefly describes some typical branching variable and branching point selection algorithms.

5.2.4.1 Branching Variable Selection

a) Largest Infeasibility Algorithm

Suppose that the solution of LP_k is $\hat{x} = \{\hat{x}_i\}_{i=1}^{n+p}$, then the *normalized infeasibility* of $\hat{x}$ with respect to each non-linear equality constraint $\zeta_j(x)$ (referred to as ζ_j-*infeasibility*) is defined as

$$U_j^{Inf}(\hat{x}) = \frac{|x_j - \zeta_j(\hat{x})|}{1 + \|\nabla\zeta_j(\hat{x})\|} \tag{5.20}$$

where $\nabla\zeta_j(\hat{x})$ is the gradient of $\zeta_j(\hat{x})$.

The *total nonlinear infeasibility* associated with each variable x_i is expressed as a linear combination of ζ_j-*infeasibility* terms,

$$\Omega_i(\hat{x}) = \mu_1 \sum_{j \in E(i)} U_j^{Inf}(\hat{x}) + \mu_2 \max_{j \in E(i)} U_j^{Inf}(\hat{x}) + \mu_3 \min_{j \in E(i)} U_j^{Inf}(\hat{x}) \tag{5.21}$$

where the set $E(i)$ denotes the indices of variables that are dependent on x_i. The parameters $\mu_k > 0$, $\forall k \in \{1, 2, 3\}$ are defined such that $\mu_1 + \mu_2 > 0$ to ensure that $\hat{x}$ is infeasible if and only if $\Omega_i(\hat{x}) > 0$ for at least one variable x_i.

The *Largest Infeasibility Algorithm* chooses to branch on the variable x_i that has the largest $\Omega_i(\hat{x})$ [11], i.e. the variable that, by the above measure, gives rise to the largest primal infeasibility. This approach is relatively inexpensive, but does not correspond to any of the surrogate criteria outlined above.

b) Strong Branching

The *strong branching* approach proposed by Applegate [8, 9] aims to find the branching variable that gives rise to the biggest improvement in the lower bound. This is achieved using a *lookahead* branching procedure that can be summarized as follows: for each branching variable candidate x_i, simulate a branch at the point $x_i^b \in [x_i^l, x_i^u]$, where $N_k \rightarrow \{N_k^- = N_k \cap \{x \mid x_i^l \leq x_i \leq x_i^b\}, N_k^+ = N_k \cap \{x \mid x_i^b \leq x_i \leq x_i^l\}\}$, then

- Find the difference between the lower bound of P_k^- and the lower bound of P_k, denote as $\psi_i^{k-} = \underline{z_k^-} - \underline{z_k}$, and
- Find the difference between the lower bound of P_k^+ and the lower bound of P_k, denote as $\psi_i^{k+} = \underline{z_k^+} - \underline{z_k}$.

The *branching score* of each variable x_i is expressed as a weighted sum of ψ_i^{k-} and ψ_i^{k+}

$$U_i^{Strong} = \alpha \max(\psi_i^{k-}, \psi_i^{k+}) + (1 - \alpha) \min(\psi_i^{k-}, \psi_i^{k+}). \tag{5.22}$$

Strong branching then chooses the branching variable with the largest U_i^{Strong}, i.e. the variable that when branched gives the largest weighted improvement in the lower bound. The weight $\alpha \in [0, 1]$ is determined empirically. For instance, in Belotti [11], α was fixed at 0.15.

Strong branching is very effective at reducing the size of the search tree, however it is also a very expensive procedure to perform [11]. For practical purposes, less expensive variants with similar attributes are required.

c) Reliability Branching

The *reliability branching* approach proposed by Achterberg [3] involves computing much less expensive heuristic equivalents of ψ_i^{k-} and ψ_i^{k+} known as *pseudocosts*. Strong branching is only performed at the first few nodes of the search tree in order to *seed* these pseudocosts.

The pseudocosts associated with a variable x_i is denoted as ϕ_i^{k-} and ϕ_i^{k+}. They are initially assigned the values of ψ_i^{k-} and ψ_i^{k+} computed using strong branching; and after that, updated using information obtained every time x_i is selected as a branching variable. Supposing that K_i^- and K_i^+ are the set of all left and right hand side node indices created from branching at x_i, then

$$\phi_i^{k-} = \frac{1}{|K_i^-|} \sum_{k \in K_i^-} \frac{\psi_i^{k-}}{\sigma_i^{k-}}, \qquad \phi_i^{k+} = \frac{1}{|K_i^+|} \sum_{k \in K_i^+} \frac{\psi_i^{k+}}{\sigma_i^{k+}}. \tag{5.23}$$

The *multipliers* σ_i^{k-} and σ_i^{k+} are scalar quantities representing the change in x_i resulting from the creation of nodes N_k^- and N_k^+, respectively. Note that ψ_i^{k-} and ψ_i^{k+} are determined *after* branching takes place, unlike in strong branching where they are determined *before* the actual decision to branch. The pseudocosts therefore represent the average of the change in the linear relaxation (ψ_i^{k-} and ψ_i^{k+}), divided by the change in the branching variable (σ_i^{k-} and σ_i^{k+}).

Suppose x_i has been chosen as the branching variable, and $x_i^b \in [x_i^l, x_i^u]$ as the branching point, such that $N_k \to [N_k^-, N_k^+]$. The subproblems of P corresponding to N_k, N_k^-, and N_k^+ are denoted as P_k, P_k^-, and P_k^+, where the linear relaxations of P_k, P_k^-, and P_k^+ are denoted as LP_k, LP_k^-, and LP_k^+. Let $\hat{x}$, $\hat{x}^-$ and $\hat{x}^+$ be the solutions of LP_k, LP_k^-, and LP_k^+, respectively. Then the following are possible definitions of σ_i^{k-} and σ_i^{k+} [11]:

- **rb-inf** (Infeasibility):

$$\sigma_i^{k-} = \sigma_i^{k+} = \Omega_i(\hat{x}) \tag{5.24}$$

  where $\Omega_i(\hat{x})$ is the infeasibility of x_i (see Equation 5.21). In other words, the change in x_i is measured by the infeasibility of the LP solution for P_k, denoted as $\hat{x}$.

- **rb-int-br** (Variable Bound):

$$\sigma_i^{k-} = x_i^b - x_i^l, \quad \sigma_i^{k+} = x_i^u - x_i^b. \tag{5.25}$$

  The change in x_i is measured by changes in its bounds due to branching. However, if x_i^l (or x_i^u) is infinite then the infeasibility of x_i is used instead, i.e. $\sigma_i^{k-} = \Omega_i(\hat{x})$ (or $\sigma_i^{k+} = \Omega_i(\hat{x})$).

- **rb-lpdist** (LP):

$$\sigma_i^{k-} = \|\hat{x} - \hat{x}^-\|_2, \quad \sigma_i^{k+} = \|\hat{x}^+ - \hat{x}\|_2. \tag{5.26}$$

  The change in x_i is measured by the distance between the solution of the new LP created after the branch, and the solution of the old LP created before the branch.

- **rb-proj** (LP-Projection):

$$\sigma_i^{k-} = \|\hat{x} - \text{Proj}[\hat{x} \to LP_k^-]\|_2, \quad \sigma_i^{k+} = \|\hat{x} - \text{Proj}[\hat{x} \to LP_k^+]\|_2. \tag{5.27}$$

  where $\text{Proj}[\hat{x} \to LP_k^-]$ is the projection of $\hat{x}$ on to LP_k^-, and $\text{Proj}[\hat{x} \to LP_k^+]$ is the projection of $\hat{x}$ on to LP_k^+. Hence, the change in x_i is measured by the distance between $\hat{x}$ and its projection on to the LP created after the branch.

5.2.4.2 Branching Point Selection

This section briefly discusses approaches for selecting a branching point x_i^b. For this purpose, we assume that the variable selected for branching, x_i, is continuous on the line of real numbers.

Supposing that $\hat{x}$ is the solution of LP_k but is infeasible in P_k, then a conceivable goal of branch selection is to ensure that the point $\hat{x}$ is excluded from any subsequent sub-problem. If x_i is selected as the branching variable, then selecting $\hat{x}_i$ as the branching point is sufficient to exclude $\hat{x}$. However, $\hat{x}_i$ may be skewed towards one of the endpoints of $x_i \in [x_i^l, x_i^u]$ creating one sub-problem that is significantly more difficult to solve than the other, which results in an unbalanced search tree. Thus it is common to select a branching point that is a convex combination of $\hat{x}_i$ and the midpoint $x_i^m = (x_i^l + x_i^u)/2$. With some additional modifications we can formulate a selection strategy that guarantees a minimum distance (b) to the bounds; this can be expressed as

$$x_i^b = \max\{x_i^l + b, \min\{x_i^u - b, \alpha\hat{x}_i + (1-\alpha)x_i^m\}\} \tag{5.28}$$

where $0 < \alpha < 1$ and $b = \beta(x_i^u - x_i^l)$ for $0 < \beta < \frac{1}{2}$. In Belotti [11] α and β are set to 0.25 and 0.2, respectively.

Furthermore, if a local minimum x^* of P_k is known then choosing x_i^* as the branching point for x_i could also be effective based on the same rationale used for aggressive feasibility based bounds tightening. Other branching point selection approaches include: the *minimum area* strategy that aims to find the point which minimizes the total area of the new convex relaxations, LP_k^- and LP_k^+, and the *balanced* strategy that aims to find the point that balances the area of LP_k^- and the area of LP_k^+ by choosing a point such that the distance between the convex relaxation and the function ($x_i = \zeta_i(x)$) is equal in both sub-problems.

5.2.5 Finding All Global Minima

A fundamental task in applied mathematics, engineering, and science involves finding all solutions to a problem. In many cases this can be posed as a problem in finding all global minimizers of an optimization problem.

A simple approach to find all global minima is to solve the global optimization problem and then solve it again with new constraints that excludes the previously found solution. This process is then repeated until no further solutions can be found. Suppose that a global minimizer $x^* \in \mathbb{R}^n$ has been found, then a valid constraint which excludes x^* is

$$\sum_{i=1}^{n}(x_i - x_i^*)^2 \geq \delta \tag{5.29}$$

where $\delta > 0$ is a small margin of separation between any new solution and x^*.

However, since there can be arbitrarily many global optima within a given region in Euclidean space, this method will not necessarily find all global minima, but instead a subset where each global minimizer is separated by at least $\sqrt{\delta}$ in Euclidean distance.

While the above approach is conceptually simple, it requires multiple executions of sBB, each time, doing a substantial amount of duplicate work. To avoid this, it is more efficient to implement the above process in the branch and bound algorithm itself, in particular, by modifying the fathoming mechanism [98, 129]. The essential modifications to the original sBB scheme provided in Section 5.2.1 is summarized as follows:

- No longer terminate the branch and bound procedure upon ϵ-convergence, as there may be other global minimizers remaining in the search tree. Instead, terminate only when there are no more nodes remaining in the search tree.
- No longer fathom a node N_k simply because LP_k is feasible, since there may be multiple global minimizers remaining within N_k. Only fathom if it can be shown that P_k is infeasible, or that P_k is sub-optimal.
- Fathom nodes that are contained within a proscribed radius around an existing global minimizer. This serves the same purpose as adding the constraint in Equation 5.29.

5.2.6 Historical Developments

Deterministic global optimization techniques were originally developed from research in combinatorial optimization and interval analysis [111]. Initial work in this area began in the early 1960s as the first electronic computers were being introduced into research environments. The advent of the electronic computer was especially important in allowing divide-and-conquer approaches such as branch and bound to be performed feasibly.

5.2.6.1 Developments from Combinatorial Optimization

In contrast to Linear Programming problems (LP), there are no known polynomial time approaches for solving combinatorial optimization problems such as *Integer Linear Programming* (ILP) and *Mixed-Integer Linear Programming* (MILP). However, one can easily derive an LP relaxation of any given ILP or MILP by dropping the requirement for integrality. The observation that it is possible

to construct more accurate linear relaxations by sub-dividing variable bound constraints subsequently led to the development of the branch and bound method.

The branch and bound method for discrete optimization problems was first introduced by Land and Doig [77], and Little [90] in the early 1960s to solve the *Travelling Salesman Problem*. The first paper concerning branch and bound in non-convex continuous optimization was by Falk and Soland [24] in 1969, where they outline a piece-wise linear relaxation applicable to some non-convex problems. In 1972, McCormick [100] introduced the now ubiquitous linear relaxation for products (Equation 5.9), which allowed general factorable problems to be solved using branch and bound.

For long periods after the 1960s, work on branch and bound in non-convex continuous global optimization were largely restricted to very problem specific applications and theoretical convergence proofs [84]. In this period, few attempts were made for instance to automate the generation of linear relaxations for general problems. It was not until the early 1990s that research in this area began to gather pace. This coincided firstly with rapid advances in the capabilities of computing hardware, and secondly with the emergence of a number of robust general-purpose (deterministic) branch and bound solvers for *mixed-integer non-linear programming* (MINLP).

The more prominent branch and bound solvers that were introduced during the 1990s included BARON (*Branch And Reduce Optimization Navigator*) by Sahinidis and co-workers [129], and αBB by Floudas and co-workers [4, 5] which uses the αBB algorithm as a basis for generating convex relaxations. These solvers, for the first time, allowed a wide range of applications to be explored using deterministic global optimization without the need to do additional work other than provide the mathematical programming formulation of the problem being considered.

5.2.6.2 Developments from Interval Analysis

In 1959, Moore and Yang [107] showed that in principle the global minimum of a function over a box can be determined with arbitrary accuracy by repeated subdivision of its domain, and repeated interval evaluation of its range. The first deterministic global optimization solvers that were able to deal with general non-convex NLPs were in fact based on interval analysis. Unfortunately, interval based approaches are generally slower to converge than linear relaxation based approaches due to the tendency for interval bounds to greatly overestimate the range of the objective function [84]. However if one desires mathematical certainty with respect to global optimality, as well as numerical certainty with respect to numerical errors, there is no alternative but to apply an interval analysis-based

approach. The most well known solver used in this area is `GlobSol` [67] developed by Kearfott and co-workers.

For further information and insight into the implementation of deterministic branch and bound solvers, and their various components, see Adjiman and Floudas [4, 5], Belotti [11], Floudas [26], Horst and Pardalos [50], and Sahinidis [129, 151] amongst others. A review of recent advances in deterministic global optimization are provided by Floudas and Gounaris [27], and Neumaier [111]. Implementation related topics are covered extensively by Liberti [163, 84]. The books by Hansen [42] and Kearfott [66] provide a view of branch and bound solvers specifically from the interval analysis perspective.

5.3 Problem Formulation

The RHF expression given in Equation 5.2 almost always involves an expansion of the molecular orbitals in terms of a *linear combination of atomic orbitals* (LCAO)

$$\phi_\mu = \sum_{a=1}^{N} c_{\mu,a}\chi_a \tag{5.30}$$

where $\{\chi_a\}_{a=1}^{N}$ are the *atomic orbitals* and $\{c_{\mu,a}\}_{a=1}^{N}$ are the *molecular orbital coefficients*. Although in principle an atomic orbital can be any arbitrary mathematical *basis function*, in practice they are usually *Contracted Gaussian functions* (CGF) formed as a linear combination of *Primitive Gaussian functions* (PGF), $\chi_{a,i}$:

$$\begin{aligned} \chi_a &= \textstyle\sum_{i=1}^{K_a} D_{p,i}\chi_{a,i}, \quad \text{where} \\ \chi_{a,i} &= (X - X_a)^{x_a}(Y - Y_a)^{y_a}(Z - Z_a)^{z_a} e^{\alpha_{a,i}|r-\mathbf{R_a}|^2} \end{aligned} \tag{5.31}$$

where K_a is the degree of contraction, $D_{a,i}$ is a *contraction coefficient*, $\mathbf{R_a} = (X_a, Y_a, Z_a)$ are the *atomic coordinates* where the atomic orbital is centered, $\mathbf{l_a} = \{x_a, y_a, z_a\}$ are the Cartesian *angular components* of the orbital, and $\{\alpha_{a,i}\}$ is the *orbital exponent*.

Using the LCAO expansion, the objective function of RHF in Equation 5.2 can be expressed formally as

$$\begin{aligned} \min \quad & 2[\textstyle\sum_{\mu=1}^{N_{elec}/2}\sum_{a=1}^{N}\sum_{b=1}^{N} c_{\mu,a}c_{\mu,b}\{(\chi_a| -\frac{1}{2}\triangle^2|\chi_b) - \sum_{A=1}^{N_{atom}}(\chi_a|\frac{Z_A}{r_{iA}}|\chi_b)\}+ \\ & \textstyle\sum_{\mu=1}^{N_{elec}/2}\sum_{v=1}^{N_{elec}/2}\sum_{a=1}^{N}\sum_{b=1}^{N}\sum_{c=1}^{N}\sum_{d=1}^{N} c_{\mu,a}c_{\mu,b}c_{v,c}c_{v,d}\{(\chi_a\chi_b|\chi_c\chi_d) - \\ & \tfrac{1}{2}(\chi_a\chi_d|\chi_c\chi_b)\}] + V_{NN} \end{aligned} \tag{5.32}$$

with the constraints of Equation 5.2 expressed as

$$s.t. \sum_{a=1}^{N} \sum_{b=1}^{N} c_{\mu,a} c_{\upsilon,b} (\chi_a | \chi_b) = \delta_{\mu\upsilon}, \quad \forall \mu, \upsilon = \{1, 2, 3, .., N_{elec}/2\} \tag{5.33}$$

where the symbols are formally defined as follows: $N_{elec} \in \mathbb{Z}$ denotes the number of electrons in the system (assumed to be even numbered); $N \in \mathbb{Z}$ denotes the number of atomic orbitals; $N_{atom} \in \mathbb{Z}$ denotes the number of atoms; the indices $\mu \in \mathbb{Z}$ and $\upsilon \in \mathbb{Z}$ are used to reference molecular orbitals; the indices $a \in \mathbb{Z}$, $b \in \mathbb{Z}$, $c \in \mathbb{Z}$, and $d \in \mathbb{Z}$ are used to reference atomic orbitals; and $\{c_{\mu,a} \in \mathbb{R} : \mu \in \{1, 2, .., N_{elec}/2\},\ a \in \{1, 2, .., N\}\}$ denotes the set of molecular orbital coefficients.

In a *point energy calculation*, which can be performed using the SCF method, the ground state energy is found by minimizing E_{HF} with respect to the set of M.O coefficients, $\{c_{\mu,a}\}$. This is a *polynomial programming* problem that can be solved using many existing solvers such as `BARON`, αBB, and `GlobSol`. In this thesis we are primarily concerned with developing methods for solving the *geometry optimization*, which also leads to further applications in Hartree-Fock theory by implication.

5.3.1 Geometry Optimization

The variation in the total energy as a result of changes in the molecular geometry is referred to as the *potential energy surface* (PES). The dimension of the PES is dependent on the number of atoms in the system. Figure 5.2 illustrates a three-dimensional PES, where the x-axis and y-axis represent the molecular geometry and the z-axis represents the corresponding total energy value.

The process of identifying stationary points on the PES is important in understanding the nature of chemical reactions and in predicting the structure of atoms and molecules [29]:

- **A local minimum** corresponds to a *stable geometry* of the system being described. In a single molecule system a stable geometry represents a structural isomer of the molecule, while in a multi-component system it represents the geometry of a set of *reactants* or a set of *products* of a reaction. In the latter case, the difference between the energy of the reactant and the energy of the product is the *heat of reaction*.
- **A saddle point** corresponds to a *transition state* that connects two stable states. In a reaction, the highest energy state along the *reaction path* is a

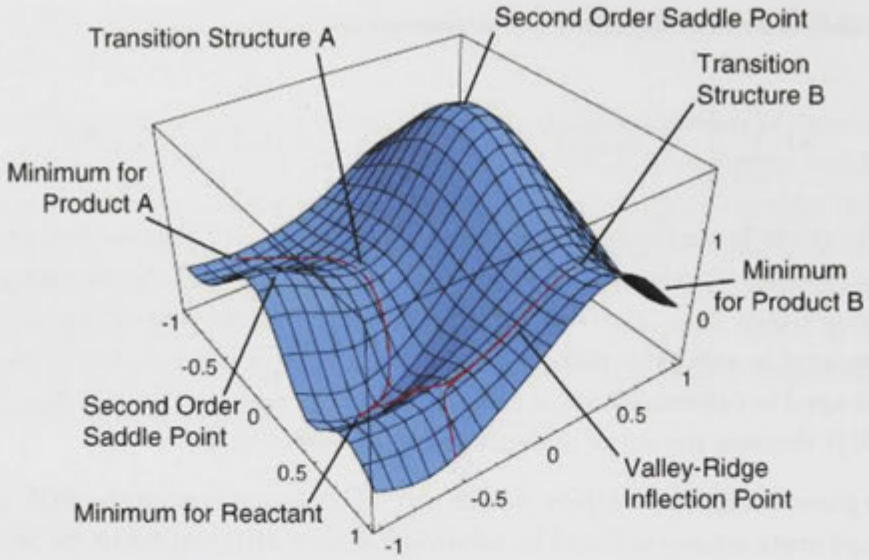


Figure 5.2: A potential energy surface. From Schlegel [135].

transition state. The difference between the energy of this transition state and the energy of the reactant can be used to predict the energy required to initiate the reaction.

Geometry optimizations are predominantly performed for the purpose of locating a local minimum on the PES which corresponds to an equilibrium geometry [29]. At the Hartree-Fock level, this is equivalent to minimizing E_{HF} with respect to the molecular orbital coefficients $\{c_{\mu,a}\}$, and at the same time, the atomic coordinates $\{\mathbf{R}_a\}$.

In conventional electronic structure codes, a geometry optimization typically begins with a starting molecular geometry specified as input. This geometry represents a point on the PES, which is then iteratively varied until a stationary point is found. Each iteration consists of two stages: the first stage involves solving the point energy problem with respect to the current point on the PES in order to obtain i) the energy, ii) the gradient, and (sometimes) iii) the Hessian matrix; the second stage determines how far and in what direction to make the next step based on the information obtained from the current point.

The iterative methods typically used for this purpose are Quasi-Newton methods, in particular, the *Davidson-Fletcher-Powell* method (DFP) and the *Broyden-Fletcher-Goldfarb-Shanno* (BFGS) method [148, 135]. A converged solution yields a stationary point on the PES, however there is no guarantee this corresponds to the point of interest. Like all local optimization procedures, a successful outcome depends, in a probabilistic sense, on finding a good starting point. This issue is

more problematic in geometry optimization than in point energy calculation, given the additional degrees of freedom involved.

5.4 Solver Implementation

The one and two-electron integrals in the geometry optimization problem are not constant since the atomic coordinates $\{\mathbf{R_a}\}$ are allowed to vary. This leads to a formulation of RHF that is significantly more difficult than that corresponding to the point energy problem. As such, we are not aware of any existing deterministic MINLP solver that is able to address this problem.

The purpose of the following section is to introduce approaches for performing geometry optimizations using a linear relaxation-based branch and bound method. We focus in particular on the central issue of generating linear relaxations for expressions involving one and two-electron integrals.

5.4.1 The Two-Electron Repulsion Integral

The two-electron repulsion integral (ERI) can be written as

$$(\chi_a\chi_b|\chi_c\chi_d) \equiv \iint \frac{\chi_a(r_1)\ \chi_b(r_1)\chi_c(r_2)\ \chi_d(r_2)dr_1dr_2}{|r_1 - r_2|} \tag{5.34}$$

$$(\chi_a\chi_b|\chi_c\chi_d) \equiv \sum_{i=1}^{K_i}\sum_{j=1}^{K_j}\sum_{k=1}^{K_k}\sum_{l=1}^{K_l} D_{a,i}D_{b,j}D_{c,k}D_{d,l}[\chi_{a,i}\chi_{b,j}|\chi_{c,k}\chi_{d,l}] \tag{5.35}$$

where the square brackets denote two-electron integrals over primitive Gaussian basis functions, or *primitive ERIs*.

In Equation 5.2, there are $O(N^4)$ ERIs (where N is the number of atomic orbitals) and $O(N^2)$ one-electron integrals (which can be expressed in a similar manner to the ERIs). It is therefore to be expected that evaluation of ERIs is the most expensive computational step in a Hartree-Fock calculation [39], and also the most important consideration when implementing an sBB solver aimed at Hartree-Fock problems.

In the rest of this section, we propose two approaches for generating linear relaxations for ERIs. The first approach is based on decomposing the ERI into simpler *reduced incomplete gamma functions* or $F_m(T)$ and then finding a linear relaxation of $F_m(T)$. The second approach is based on a decomposition into

generalized contracted $(ss|ss)$-type ERIs with two atomic centers, and then finding a linear relaxation of the $(ss|ss)$ ERIs using numerically determined parameters.

5.4.2 Approach 1: Linear Relaxation of the $F_m(T)$ Expression

While there are no analytical approaches for evaluating ERIs, there are several efficient numerical approaches for computing ERIs including those by McMurchie-Davidson [103], Obara-Saika [115], and Head-Gordon-Pople [44]. Each of these approaches use recursion relations where the primitive ERI is decomposed into a linear combination of simpler integrals called the *reduced incomplete gamma function* or $F_m(T)$ for short (see Section 2.2.6)

$$F_m(T) = \int_0^1 t^{2m} e^{-Tt^2} dt \tag{5.36}$$

where $T \in \mathbb{R}$ is a non-negative variable dependant on the distance between atomic centers and the values of the Gaussian exponents. $m \in \mathbb{Z}$ is a parameter dependant on the total angular momentum, and typically ranges from 0 up to 16.

Like the ERI, there are no analytical approaches for evaluating $F_m(T)$, although it can be evaluated accurately and efficiently using a combination of series evaluation and numerical interpolation as outlined previously in Section 3.3.

Proposition 5.4. *$F_m(T)$ is monotonically decreasing, and convex for all $T \geq 0$.*

Proof. The first derivative of of $F_m(T)$ with respect to T follows from the relation

$$\begin{aligned} \frac{d}{dT} F_m(T) &= \frac{d}{dT} \int_0^1 t^{2m} e^{-Tt^2} dt \\ &= -\int_0^1 t^{2m+2} e^{-Tt^2} dt \\ &= -F_{m+1}(T). \end{aligned} \tag{5.37}$$

Since $F_m(T)$ is non-negative for all values of $m \in \mathbb{Z}$ and $T \geq 0$ as seen clearly in Equation 5.36, the gradient of $F_m(T)$ must be negative. Thus proving that $F_m(T)$ is a monotonically decreasing function.

Likewise, the convexity of $F_m(T)$ can be proven by taking the second derivative of $F_m(T)$ with respect to T using the above relation. □

Because of its convexity, it is relatively straightforward to define a linear relaxation for $F_m(T)$. Suppose we have an equality constraint of the form $z =$

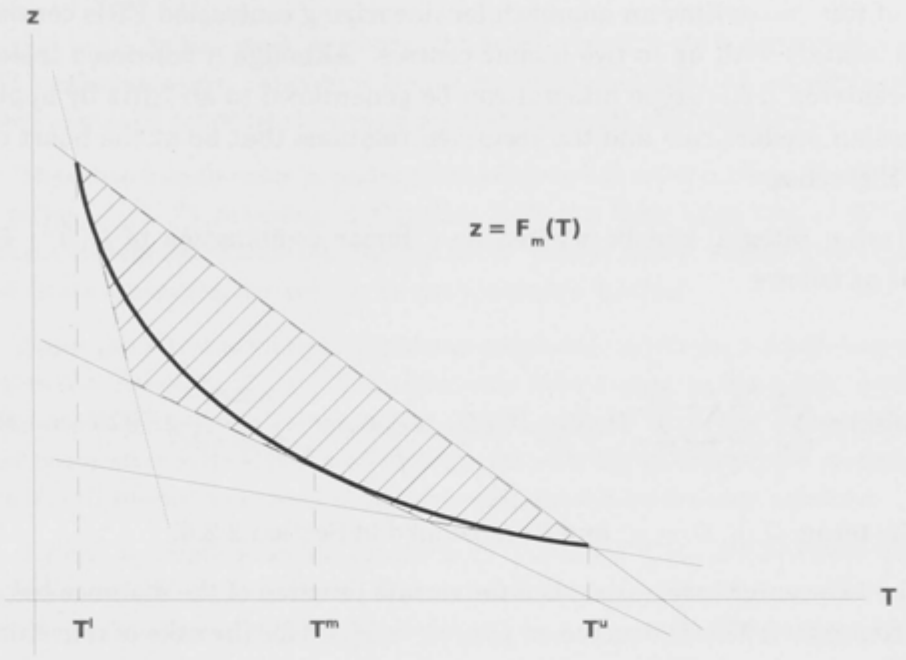


Figure 5.3: Linear relaxation scheme for $z = F_m(T)$.

$F_m(T)$, $T \in [T^l, T^u]$, a linear relaxation consisting of four linear inequalities can be expressed as

$$\begin{aligned} z &\geq -F_{m+1}(T^l)(T - T^l) + F_m(T^l) \\ z &\geq -F_{m+1}(T^m)(T - T^m) + F_m(T^m) \\ z &\geq -F_{m+1}(T^u)(T - T^u) + F_m(T^u) \\ z &\leq \frac{F_m(T^u) - F_m(T^l)}{T^u - T^l}(T - T^l) + F_m(T^l) \end{aligned} \tag{5.38}$$

where $T^m = (T^l + T^u)/2$ denotes the midpoint between T^l and T^u. This linear relaxation is illustrated in Figure 5.3. It shows that the curve is bounded below by the tangents of $F_m(T)$ at the left endpoint $T = T^l$, midpoint $T = T^m$, and right endpoint $T = T^u$. The curve is also bounded above by a line segment extending from the left to the right endpoint. Note that the gradient of $F_m(T)$ is derived from Equation 5.37.

5.4.3 Approach 2: Linear relaxation of $(ss|ss)$-type Integrals

A major concern about Approach I is the large numbers of linear inequalities which result from first decomposing each $O(N^4/8)$ integral into at least $O(K^4)$ $F_m(T)$ expressions (due to contraction) and then replacing each $F_m(T)$ with four inequalities. A natural progression is to consider ways to directly linearize individual ERIs. On

account of this, we outline an approach for linearizing contracted ERIs consisting of s-type orbitals with up to two atomic centres. Although a restricted instance, the two-centered $(ss|ss)$-type integral can be generalized to all ERIs by applying the Gaussian product rule and the recursion relations that lie at the heart of all modern ERI codes.

An $(ss|ss)$ integral can be written as a linear combination of $O(K^4)$ $F_0(T)$ functions as follows

$$(ss|ss) = \sum_{i=1}^{K_a}\sum_{j=1}^{K_b}\sum_{k=1}^{K_c}\sum_{l=1}^{K_d} D_{a,i}D_{b,j}D_{c,k}D_{d,l}G_{AB}G_{CD}\frac{2\pi^{5/2}}{\zeta\eta(\zeta+\eta)^{1/2}}F_0(T) \tag{5.39}$$

where the terms G_{AB}, G_{CD}, ζ, and η are defined in Section 2.2.6.

In the two-center case, $(ss|ss)$ is a univariate function of the distance between the two centers (r). This is denoted as $f_{eri}(r), r \in [r^l, r^u]$ for the sake of convenience.

Proposition 5.5. *$f_{eri}(r)$ is a monotonically decreasing function with at most one inflection point, $r_{inf} \geq 0$. Furthermore, $f_{eri}(r)$ is concave for $r \leq r_{inf}$, and convex for $r \geq r_{inf}$.*

Remark 5.6. The monotonicity of $f_{eri}(r)$ with respect to r follows intuitively from the definition of a two-electron repulsion integral.

The existence of an inflection point r_{inf} can be derived by applying the *intermediate value theorem* with respect to $f''_{eri}(r)$ at $r = 0$ and at $r = N_0$, where $N_0 \geq 0$ is a sufficiently large number. The uniqueness of r_{inf} on the other hand has not yet been proven in the general instance. However, it can be established in specific instances by showing that there is exactly one $r_{inf} \geq 0$ such that $f''_{eri}(r_{inf}) = 0$. Following an approach proposed by Maranas [98], this can be reformulated as a global optimization problem by introducing a slack variable $s \in \mathbb{R}$

$$\begin{aligned} \min \quad & s \\ \text{s.t. } & f''_{eri}(r) - s \leq 0 \\ & -f''_{eri}(r) + s \leq 0 \\ & r \geq 0 \\ & s \geq 0 \end{aligned} \tag{5.40}$$

so that the objective $s = 0$ if and only if the minimizer corresponds to one of the roots of $f''_{eri}(r)$. Therefore if there is only one global minimizer, there can only be one unique root for $f''_{eri}(r)$. Such a proof can be carried out by either using i) an interval analysis based root finding method, or ii) the modified variant of sBB discussed in

Section 5.2.5. In this work we choose the latter approach as it reuses the existing sBB implementation, however it should be mentioned that the interval analysis based approach would be more rigorous with respect to rounding error.

Note also that in order to perform the above proof, we must first expand $f''_{eri}(r)$ in terms of $F_m(T)$ according to Equation 5.39 and then apply the $F_m(T)$ linear relaxation scheme outlined in Section 5.4.2. Hence, this is an example of where one linear relaxation scheme can be used to derive another.

Since the above is an optimization problem only involving a single non-linear univariate function, $f''_{eri}(r)$, it is relatively inexpensive to formulate proofs of uniqueness for all $(ss|ss)$ integrals in a given system. From our observation, the time required to solve Equation 5.40 using the sBB approach typically ranged from less than 1 second, to around 5 seconds using a standard desktop computer.

Finally, by implication, the concavity of $f_{eri}(r)$ for $r \leq r_{inf}$ can be demonstrated by evaluating $f''_{eri}(r)$ for any $r \in [0, r_{inf})$, while the convexity of $f_{eri}(r)$ for $r \geq r_{inf}$ can be shown by evaluating $f''_{eri}(r)$ for any $r \in (r_{inf}, \infty)$. Thus, the sufficient conditions for the existence and uniqueness of an inflection point r_{inf} can be established, albeit on a case by case basis, and subject to numerical errors in the modified sBB solver.

For the purpose of the discussion below, we define r_a and r_b to be significant points where the tangent of $f_{eri}(r)$ passes through the endpoints $(r^l, f_{eri}(r^l))$, and $(r^u, f_{eri}(r^u))$, respectively. Providing that the above proposition is satisfied, both r_a, and r_b can be determined reliably by applying the bisection method to the following equations

$$\frac{f_{eri}(r_a) - f_{eri}(r^l)}{r_a - r^l} - f'_{eri}(r_a) = 0 \tag{5.41}$$

and

$$\frac{f_{eri}(r^u) - f_{eri}(r_b)}{r^u - r_b} - f'_{eri}(r_b) = 0. \tag{5.42}$$

The form of $f_{eri}(r)$ is not unlike that of an odd-degree monomial (x^{2n+1}) for which a linear relaxation scheme has been established previously by Liberti and Pantelides [87]. Given an equality constraint of the form $z = f_{eri}(r), r \in [r^l, r^u]$, the linear relaxation scheme that is applied depends on the value of the endpoints (r^l and r^u) in relation to r_{inf}, r_a and r_b. This gives rise to five unique cases.

As a characteristic example, we consider **Case I** where r_{inf}, r_a, and r_b are all within $[r^l, r^u]$. The linear relaxation for this case is illustrated in Figure 5.4 and

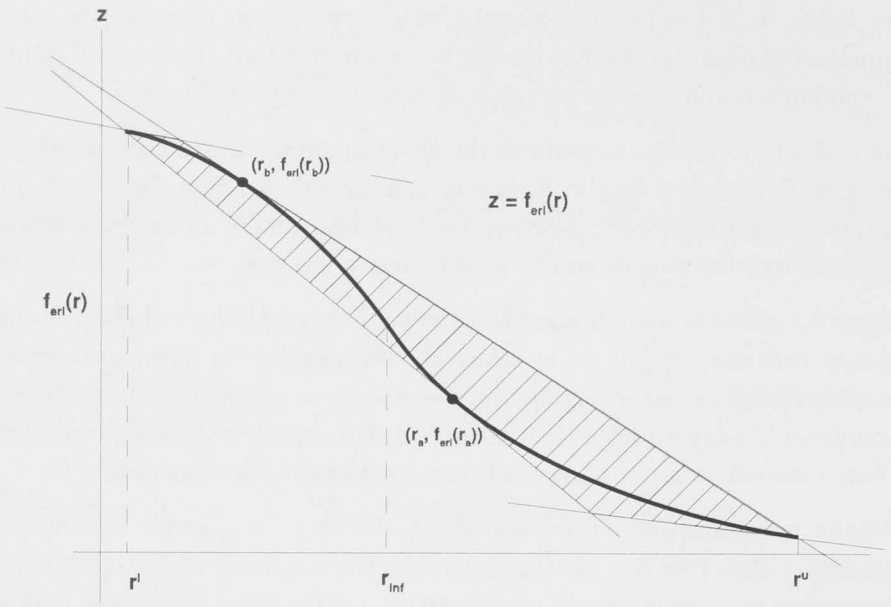


Figure 5.4: Case I: Linear relaxation scheme for $z = f_{eri}(r)$, where $r^l \leq r_b, r_{inf}, r_a \leq r^u$

consists of four linear inequalities

$$\begin{aligned} z &\geq f'_{eri}(r^u)(r - r^u) + f_{eri}(r^u) \\ z &\geq f'_{eri}(r_a)(r - r_a) + f_{eri}(r_a) \\ z &\leq f'_{eri}(r^l)(r - r^l) + f_{eri}(r^l) \\ z &\leq f'_{eri}(r_b)(r - r_b) + f_{eri}(r_b) \end{aligned} \tag{5.43}$$

where $f'_{eri}(r)$ is the first derivative of $f_{eri}(r)$ with respect to r. The curve is bounded below by the tangents of $f_{eri}(r)$ at $r = r_a$ and at $r = r^u$, and bounded above by the tangents of $f_{eri}(r)$ at $r = r^l$ and $r = r_b$.

For the other cases, the linearization is applied as follows and as illustrated in Figure 5.5. Note that r^m is the midpoint between r^l and r^u, i.e. $r^m = (r^l + r^u)/2$.

- **Case II:** $r^l \leq r_b, r_{inf} \leq r^u$ but $r_a > r^u$

$$\begin{aligned} z &\geq \frac{f_{eri}(r^u) - f_{eri}(r^l)}{r^u - r^l}(r - r^l) + f_{eri}(r^l) \\ z &\leq f'_{eri}(r^l)(r - r^l) + f_{eri}(r^l) \\ z &\leq f'_{eri}(r_b)(r - r_b) + f_{eri}(r_b). \end{aligned} \tag{5.44}$$

- **Case III:** $r^l \leq r_{inf}, r_a \leq r^u$ but $r_b < r^l$

$$\begin{aligned} z &\geq f'_{eri}(r_a)(r - r_a) + f_{eri}(r_a) \\ z &\geq f'_{eri}(r^u)(r - r^u) + f_{eri}(r^u) \\ z &\leq \frac{f_{eri}(r^u) - f_{eri}(r^l)}{r^u - r^l}(r - r^l) + f_{eri}(r^l). \end{aligned} \tag{5.45}$$

- **Case IV (Convex):** $r_{inf} \leq r^l$

$$\begin{aligned} z &\geq f'_{eri}(r^l)(r - r^l) + f_{eri}(r^l) \\ z &\geq f'_{eri}(r^m)(r - r^m) + f_{eri}(r^m) \\ z &\geq f'_{eri}(r^u)(r - r^u) + f_{eri}(r^u) \\ z &\leq \frac{f_{eri}(r^u) - f_{eri}(r^l)}{r^u - r^l}(r - r^l) + f_{eri}(r^l). \end{aligned} \tag{5.46}$$

- **Case V (Concave):** $r^u \leq r_{inf}$

$$\begin{aligned} z &\geq \frac{f_{eri}(r^u) - f_{eri}(r^l)}{r^u - r^l}(r - r^l) + f_{eri}(r^l) \\ z &\leq f'_{eri}(r^l)(r - r^l) + f_{eri}(r^l) \\ z &\leq f'_{eri}(r^m)(r - r^m) + f_{eri}(r^m) \\ z &\leq f'_{eri}(r^u)(r - r^u) + f_{eri}(r^u). \end{aligned} \tag{5.47}$$

Case II represents the case where r_{inf} is contained in $[r^l, r^u]$, but r_a is greater than r^u. Hence, the curve is bounded below by the line segment extending from the left endpoint to the right endpoint of $f_{eri}(r)$, and bounded above by the tangents of $f_{eri}(r)$ at $r = r^l$ and at $r = r_b$. **Case III** represents the case where r_{inf} is contained in $[r^l, r^u]$, but r_b is less than r^l. Hence, the curve is bounded below by the tangents of $f_{eri}(r)$ at $r = r_a$ and at $r = r^u$, and bounded below by the line segment extending from the left endpoint to the right endpoint of $f_{eri}(r)$. **Case IV** and **Case V** represents the entirely convex and concave cases, respectively.

It should be noted that, since $f_{eri}(r)$ is a twice-differentiable non-convex function, an alternative approach to the above is to use the αBB convex relaxation scheme (see Section 5.2.2.1). For example, an equality constraint $z = f_{eri}(r)$ can be replaced by two inequality constraints $z \geq L^+_{eri}(x)$ (lower bounding) and $z \leq -L^-_{eri}(x)$ (upper bounding), where $L^+_{eri}(x)$ and $L^-_{eri}(x)$ are the αBB convex relaxation of $f_{eri}(r)$ and $-f_{eri}(r)$, respectively. Figure 5.6 illustrates an αBB convex relaxation of $f_{eri}(r)$. It is visually evident that even the tightest possible αBB relaxation is not likely to be as tight as the linear relaxation proposed in this section.

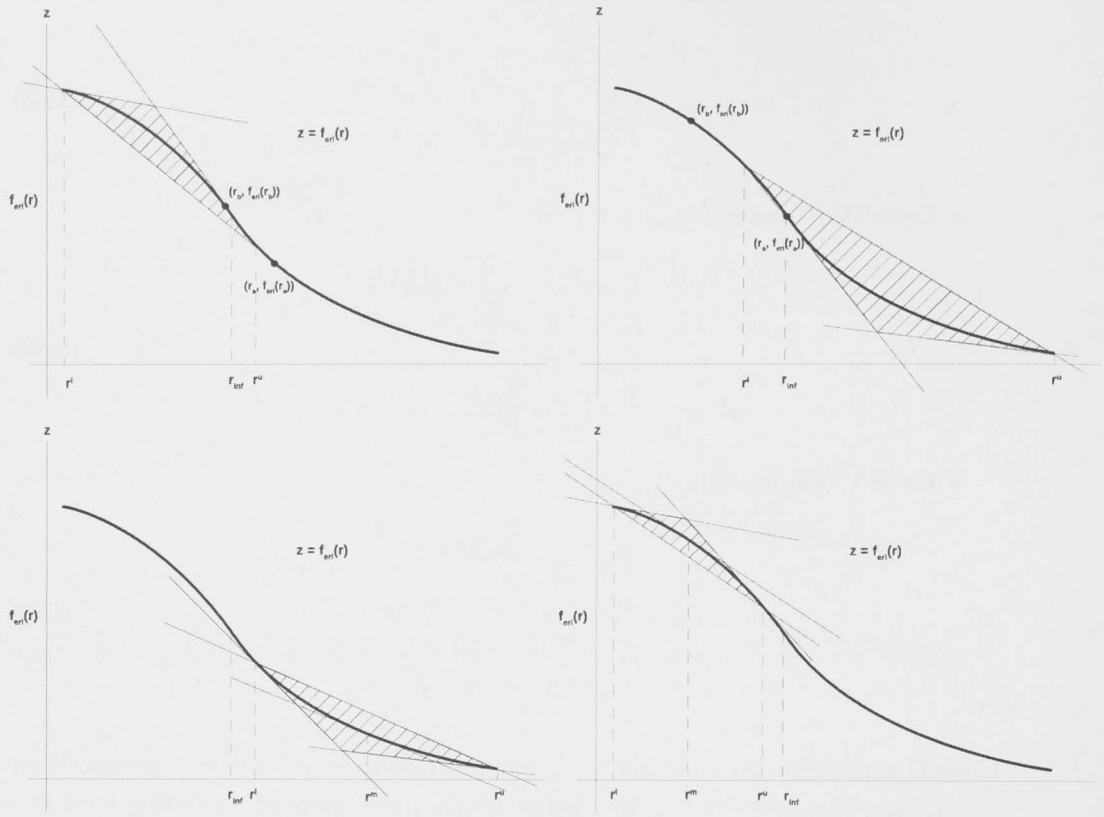


Figure 5.5: Linear relaxation scheme for $z = f_{eri}(r)$, (*top-left*) **Case II:** $r^l \leq r_b, r_{inf} \leq r^u$ but $r_a > r^u$, (*top-right*) **Case III:** $r^l \leq r_{inf}, r_a \leq r^u$ but $r_b < r^l$, (*bottom-left*) **Case IV:** $r_{inf} \leq r^l$, (*bottom-right*) **Case V:** $r^u \leq r_{inf}$

5.4.4 Implementation Details

The approaches proposed in this chapter were implemented in COUENNE (*Convex Over and Under ENvelopes for Nonlinear Estimation*) a general-purpose (deterministic) MINLP solver. Unlike the other solvers mentioned previously, COUENNE is open source, and structured so that it can be easily adapted towards solving specific applications by anyone with a working knowledge of C++ [11]. COUENNE is distributed under the *Eclipse Public Licence 1.0* and integrated with the wider COIN-OR (*Computational Infrastructure for Operations Research*) framework [92], allowing COUENNE to utilize a number of well established packages from within the same framework. For example, BONMIN provides the interface and algorithmic framework [12], routines from CBC [30] are used to implement sBB, P_k is solved by IPOPT [156], and LP_k is solved by CLP [31]. In terms of performance and robustness, COUENNE is said to be comparable to BARON, which is widely regarded to

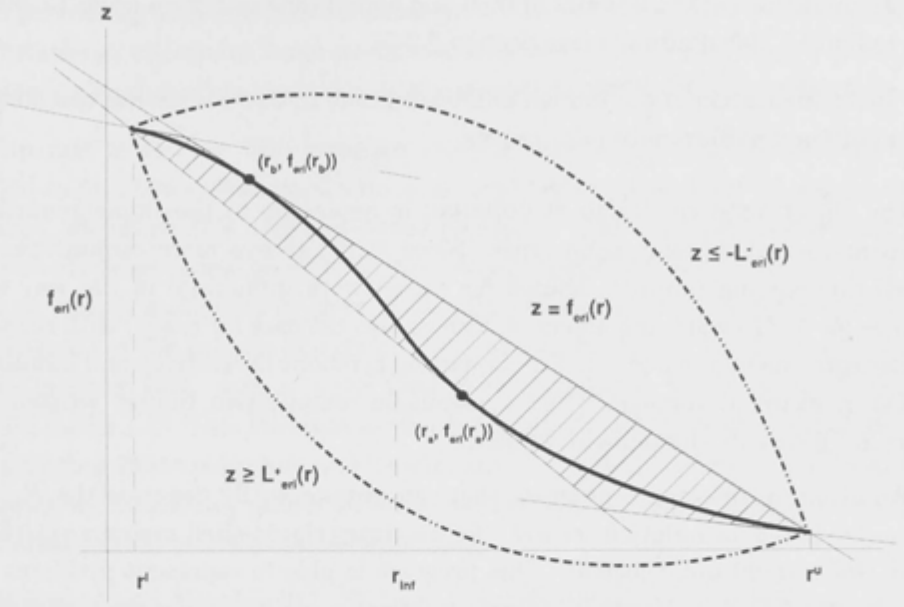


Figure 5.6: Convex relaxation of $f_{eri}(r)$ generated via αBB (dashed lines), compared to the linear relaxation generated in **Case I**.

be the benchmark amongst contemporary deterministic MINLP solvers [11, 112].

Our implementation, referred to as the *Hartree-Fock enabled* COUENNE, was written entirely in C++, using as a basis COUENNE-1.0 (revision 65), but also incorporating several important bug fixes applied to later versions of COUENNE. Significant additions and modifications to the existing code base of COUENNE and other COIN-OR packages include:

- C++ classes defining the $F_m(T)$ and $(ss|ss)$ function types. And implementation of associated routines for evaluating the function value, derivatives, range, and inverses. These routines are necessary, amongst other things, for performing bounds tightening, branch selection, and linear relaxation on expressions containing these functions.
- Code for reading an input file that specifies inflection points (r_{inf}), and symbolic derivatives for each $(ss|ss)$ function in the problem.
- Routines for reformulating and linearizing expressions involving $F_m(T)$ and $(ss|ss)$. This is based on the approaches outlined in Section 5.4.
- Routines for selecting branching points on auxiliary variables defined by $F_m(T)$ or $(ss|ss)$ functions.

- Modifications of the original branch and bound code in CBC in order to use it to find all global minima (see Section 5.2.5).
- Instrumentation of the branch and bound code in CBC so that we can obtain additional performance parameters.

The object oriented design of COUENNE is amenable to the implementation of new mathematical expression types. Since $F_m(T)$ shares many similarities to $\exp(x)$, the existing routines defined for $\exp(x)$ were influential in the way the routines for $F_m(T)$ were implemented. This is also the case for $(ss|ss)$ with respect to odd-degree monomials x^{2n+1}. For computing symbolic derivatives and handling file I/O, a modified version of Ev3 (a symbolic computation library written by Liberti [82]) was also linked with COUENNE.

We also implemented a program that can automatically generate the $F_m(T)$ and $(ss|ss)$ based formulations of RHF for arbitrary closed-shell systems specified in the *Gaussian03* input format. This program is able to represent problems in both the GAMS [15] and AMPL [32] mathematical modelling languages, with the latter being accepted by COUENNE. Ev3 is again used to construct an in-memory representation of the problem and other intermediate quantities.

Unlike non-deterministic approaches, the process used in deriving an ϵ-optimal solution is as critical as the global optimality of the solution itself. An erroneously implemented solver can, on many occasions, find the correct solution, but may not be rigorous in the process in which it establishes ϵ-convergence. From our experience, these errors are often caused by linear relaxations or bounds tightenings that mistakenly eliminate un-searched regions of the domain. There are many ways in which these type of mistakes can be made, identifying and then rectifying them is a difficult task. Furthermore, since implementation related issues are not at all transparent during run time, it often requires a very specific instance of failure for them to become apparent. For these reasons, the implementation of deterministic global optimization code requires sound software engineering practices [163].

The source code for *Hartree-Fock enabled* COUENNE and other programs mentioned in this section are available on the author's web site [56]. Additional work may be required to integrate the former with more recent versions of COUENNE.

5.5 Related Work

Given the immense computational costs involved in deterministic global optimization, it is not surprising that its application has so far been restricted to relatively

small problems, with few notable exceptions (see [27]). Previous works relating to computational chemistry have predominantly centered on finding the solution to simple empirical models that do not take quantum effects into consideration.

Lin and Stadtherr [89] used an interval analysis-based branch and bound method to find the global minimum of several types of empirical potential energy surfaces, including the Lennard-Jones potential. The problems considered were particularly interesting because of the large numbers of local minima that were present, which are likely to pose difficulties to iterative descent methods. The performance of the branch and bound code was improved by employing the Taylor model method [95], and a novel linear programming based interval Newton method (LISS_LP). This followed earlier work by Lavor [78] which employed a less sophisticated interval branch and bound code. Lin [88] also considered the problem of finding all stationary points on a Lennard-Jones potential energy surface using the same interval analysis-based approach.

Maranas and Floudas [97, 26] used a convex relaxation approach to optimize the structure of Lennard-Jones and Morse clusters. The convex relaxation was constructed using a procedure which appears to be a direct predecessor of the αBB algorithm. The optimization itself was performed using the GOP algorithm first introduced by Floudas and Visweswaran [28, 155]. Small clusters of up to 7 particles were optimized, while tight lower and upper bounds were computed for larger clusters of up to 25 particles. Floudas and his group also explored a number of other chemistry related applications using deterministic global optimization techniques. This included protein folding [7], reactor network synthesis [22], multi-component design [96], and phase equilibrium problems [102], amongst others.

Empirical models are useful in providing an initial description of a particular chemical system, but are not accurate enough to provide qualitative results. It is therefore arguable whether there is any merit in applying a rigorous approach to solve what are essentially inaccurate models. It would be more appropriate to instead apply deterministic global optimization towards solving semi-empirical or ab-initio models which are more closely derived from theoretical principles.

The first published work that directly deals with problems in ab-initio quantum chemistry, and Hartree-Fock theory, was by Lavor et. al. [79]. In this work, the Hartree-Fock ground state energy of Helium and Beryllium was calculated using an interval analysis-based branch and bound method. The focus has since shifted from the interval analysis approach to the more efficient convex relaxation based approaches in later works by Lavor, Maculan, and Liberti [80, 86], and Cafieri [17].

Since the Hartree-Fock point energy problem is an instance of polynomial

programming, there are a number of approaches in literature that can be drawn upon to improve the effectiveness of the convex relaxation step. Liberti [86] applied a variation of the Reformulation Linearization Technique (RLT) first described by Sherali [141, 140]. RLT produces tighter relaxations by introducing new equality and inequality constraints formed by multiplying together the existing set of variable bound constraints and equality constraints. The main challenge in its implementation is how to manage the trade-off between having tighter relaxations and the computational costs of having many new constraints. Liberti demonstrates that RLT can reduce the computational cost in point energy calculations by one to two orders of magnitude in Helium and Beryllium test cases. A similar approach is currently being pursued by Zorn and Sahinidis [166].

Cafieri [17] compared four different approaches for generating linear relaxations for quadrilinear terms of the form $\zeta_i(x) = x_1x_2x_3x_4$. This included the decomposition of $\zeta_i(x)$ into permutations of i) three bilinear terms e.g. $y_1 = x_1x_2$, $y_2 = x_3x_4$, $\zeta_i(x) = y_1y_2$, and ii) a trilinear and a bilinear term e.g. $y_1 = x_1x_2x_3$, $\zeta_i(x) = y_1x_4$. Computational experiments comparing the performance of each of these approaches were presented. This included test cases involving the point energy calculation of Beryllium and Neon under Hartree-Fock theory.

The work presented in this chapter differs from the above body of work in that we address Hartree-Fock theory in the general instance, which allows us to consider applications beyond point energy calculations. Whereas the point energy problem is an instance of polynomial programming which is common to other well established areas of application, the problem instances considered here are specific to quantum chemistry.

Another quantum chemistry related development is the work by Fukuda and Zhao on performing electronic structure calculations using the variational two electron reduced density matrix (RDM) method [35, 165]. Unlike the Hartree-Fock equations which are generally non-convex, the RDM method gives rise to a large scale semi-definite programming problem (SDP). Since SDP is an instance of convex programming, the RDM method can, in principle, be solved reliably using local optimization. The main difficulty, however, lies in the intractability of the particular SDP associated with this problem, meaning that it is only practical to solve a lower bounding dual formulation, which can also be expressed as an SDP. Point energy calculations were performed with respect to different variants of the dual formulation, yielding results for 24 different systems, the largest being CH_3N using the STO-6G basis set [35].

5.6 Conclusions and Future Work

This chapter outlines the design and implementation of a deterministic global optimization solver for evaluating problems in Hartree-Fock theory. The approach adopted is based on the widely used branch and bound framework, which also incorporates algorithms for i) reformulating and generating linear relaxations for the primal problem, ii) variable bounds tightening, and iii) branching variable and branching point selection.

The most significant contribution of this chapter is the development of approaches for generating linear relaxations for one and two-electron integrals. Two contrasting approaches were outlined: one based on decomposing the integrals into a linear combination of $F_m(T)$ functions and then generating a linear relaxation for each $F_m(T)$, the other based on generating a linear relaxation for the $(ss|ss)$ type two-electron integrals using numerically determined parameters. These approaches were implemented as an extension of COUENNE, an open source, general purpose, deterministic MINLP solver.

In implementing these approaches in COUENNE, we were able to leverage the robustness of a general purpose MINLP solver, as well as the existing code base for performing branch and bound, bounds tightening, and branch selection. To implement a new solver from scratch that was as robust and as efficient in all of these aspects would have been prohibitively time consuming. For example, the branch and bound code implemented in CBC alone consists of over 50,000 lines of C++ code. The solver implemented provides a robust platform from which to further specialize deterministic global optimization techniques towards other applications in electronic structure theory.

In conclusion, the approaches developed in this chapter provide a way to perform Hartree-Fock calculations with strong, near analytical, guarantees on optimality. Although more computationally pragmatic approaches such as local optimization and stochastic global optimization exist, they cannot provide the same level of rigour. In the next chapter, we explore the application of deterministic global optimization to evaluating the Hartree-Fock equation with respect to i) point energy calculation, ii) geometry optimization, iii) basis set optimization, and iv) excited state calculation

Chapter

6

Application of Deterministic Global Optimization in Hartree-Fock Theory

Contents

6.1 Introduction

Despite the rapid advances made in electronic structure theory and computation, the fundamental task of finding the optimal energy state of an atomic or molecular system remains a process in which success cannot be guaranteed. In situations

where conventional local optimization based approaches fail to yield the correct result, it is usually the task of a human analyst to make informed, but often hopeful, adjustments to the underlying parameters of the problem [29], from which another attempt at finding the solution will be made. This is still an art that requires the analyst to have in-depth knowledge of both the problem that is being solved and the solver that is being used to solve it.

The previous chapter outlined the efforts made to address this issue. It centered upon the treatment of the Hartree-Fock equations as a global optimization problem, which can then be solved reliably using deterministic global optimization techniques. Even though this constitutes a substantial increase in computational cost over the already expensive approaches used in this area, the fact that it is able to obtain mathematically rigorous solutions is unique. While it remains a long term goal to use this approach to solve large scale molecular systems, the high computational costs inherent in deterministic search currently restrict the application towards model systems. For general non-convex NLPs, problems with as little as 10 to 15 degrees of freedom are still considered to be challenging even to the most advanced deterministic MINLP solvers currently available.

Working with these limitations, and conscious of the potential for future work, the aim of this chapter is to highlight the applications of deterministic global optimization in Hartree-Fock theory. Proof of concept calculations are performed in application areas including i) point energy calculation, ii) geometry optimization, iii) basis set optimization, and iv) excited state calculation. Further contributions include

- Performance characterization of COUENNE and comparable deterministic MINLP solvers in evaluating benchmark non-linear programming, and point energy calculation problems.
- The inclusion of symmetry-breaking constraints in the formulation of the Hartree-Fock equations.
- A performance characterization of the Hartree-Fock enabled COUENNE program for geometry optimization. Evaluating the impact of factors such as the choice of bounds tightening, linear relaxation, and branch selection algorithm, and the impact of varying variable bound constraints.

This chapter is organized into the following sections: Section 6.2 outlines the default experimental platform and parameters; Section 6.3 investigates the performance of COUENNE and BARON in solving a series of non-linear programming test problems; Section 6.4 presents the results for point energy calculations

performed using COUENNE, BARON, and GlobSol, and discusses ways to improve performance by removing symmetric solutions from the problem formulation; Section 6.5 presents the results for geometry optimization, comparing the performance of the $F_m(T)$ relaxation approach to the $(ss|ss)$-type integral relaxation approach, as well as other factors including the choice of branch selection algorithm; Section 6.6 presents the results for basis set optimization; Section 6.7 presents the results for excited state calculations; and finally Section 6.8 concludes the chapter with a discussion of implications and future work.

6.2 Default Experimental Platform, Parameters, and Nomenclature

All experiments are performed on a 2.13 GHz Intel Core 2 Duo 6400 processor with a total of 4GB of main memory, using 64-bit Linux 2.6.28-15.

For the problems involving non-constant one and two electron integrals, such as geometry optimization, we use the modified COUENNE implementation described in Section 5.4.4. For other problems the more recent COUENNE-3.0 (revision 516) is used. Each version of COUENNE was compiled using gcc-4.3.3 and accessed using AMPL Student Version 20091123.

The experiments consider a number of different configurations of COUENNE, each employing bounds tightening and branch selection algorithms of varying degrees of sophistication. These configurations are summarized as follows:

	Bounds Tightening	**Branching**
no-bt	None	Largest infeasibility
fbbt	FBBT	Largest infeasibility
obbt	OBBT(6)	Largest infeasibility
afbbt	FBBT/AFBBT(6)	Largest infeasibility
strong	FBBT/OBBT(0)/AFBBT(1)	Strong branching

Bounds tightening methods were discussed in detail in Section 5.2.3. Feasibility based bounds tightening (FBBT) is used in every configuration except for *no-bt* and *obbt*. Optimality based bounds tightening (OBBT) is used only in *obbt* and *strong*, with the value of γ^{OBBT} reported in parentheses. Aggressive feasibility based bounds tightening is used in *afbbt* and *strong*, with the value of γ^{AFBBT} reported in parentheses. The *fbbt*, *obbt*, and *afbbt* configurations use the largest infeasibility branching algorithm discussed in Section 5.2.4.1, while the *strong* configuration uses the more advanced strong branching-based reliability

branching approach also discussed in Section 5.2.4.1. *fbbt* and *afbbt* are identical configurations but for the use of aggressive feasibility based bounds tightening in *afbbt*. *no-bt* does not use any bounds tightening at all.

We use the label A1 - $F_m(T)$ to refer to the $F_m(T)$ decomposition approach outlined in Section 5.4.2 while the label A2 - $(ss|ss)$ is used to refer to the $(ss|ss)$ decomposition approach outlined in Section 5.4.3.

As a reference, we also use the following general purpose deterministic MINLP solvers:

- BARON version 9.0.7, accessed using GAMS version 23.6.
- GlobSol downloaded in February 2011 [64], accessed using FORTRAN90.

BARON is a commercial deterministic MINLP solver that has many similarities to COUENNE, especially in the way it performs branch and bound, bounds tightening, branch selection, linear relaxation, etc. BARON is also considered to be the benchmark amongst contemporary MINLP solvers both in terms of performance and reliability [112].

GlobSol is a deterministic global optimization solver based on interval analysis. Although not particularly fast compared to other solvers, it is more reliable in that it is able to compute bounds on the global minimum notwithstanding rounding errors or numerical instability. This feature is not available to BARON, COUENNE and most other well known convex relaxation based branch and bound solvers.

The execution time is limited to 36 hours for geometry optimization experiments and 2 hours for all other experiments. In all quantum chemistry related optimizations, the *absolute* ϵ-convergence threshold is set to 1KCal/Mol or approximately 1.6×10^{-3} Hartrees. This threshold corresponds to generally accepted notions of *chemical accuracy*. In all other optimizations, the *relative* ϵ-convergence threshold is set to $\epsilon = 1 \times 10^{-3}$.

6.3 Non-Linear Programming Benchmark

The following experiment evaluates the performance of COUENNE and BARON for solving a suite of NLP test problems obtained from the GLOBALLib benchmark [1]. These problems provide a useful point of reference for the Hartree-Fock problems to be considered later on, since they both belong to similar classes of NLPs (multivariate polynomial programming).

Problem	*var*	*aux*	*lin*	*non-lin*
camshape100	199	499	201	198
ex3_1_1	8	20	7	5
ex3_1_2	5	20	7	8
ex3_1_3	6	28	14	8
ex5_4_4	27	65	29	21
ex7_3_4	12	45	19	19
ex7_3_5	13	43	22	17
ex8_4_4	17	66	31	30
ex8_4_7	62	223	101	100
hhfair	27	63	35	16
meanvar	7	37	3	28
qp1	50	1327	3	1275
qp2	50	1327	3	1275
qp3	100	152	53	50
st_m1	20	52	12	20
st_m2	30	82	22	30
st_qpk1	2	10	5	3
st_qpk2	6	24	7	11
st_qpk3	11	44	12	21

Table 6.1: Size and complexity of NLP benchmark problems.

Table 6.1 summarises the size and complexity of the problems considered in this experiment; *var* denotes the number of variables in the original problem, P; *aux* denotes the number of auxiliary variables in the standardized problem, $\tilde{P}$; *lin* denotes the number of linear constraints in $\tilde{P}$; and *non-lin* denotes the number of non-linear constraints in $\tilde{P}$. These metrics will be used to describe problems throughout this chapter.

The number of variables (*var*) is related to the degrees of freedom in the original problem. The number of auxiliary variables (*aux*) is related to the number of factorable terms in the original problem. By definition, the number of auxiliary variables (*aux*) should be equal to the sum of the number of variables (*var*), the number of linear constraints (*lin*), and the number of non-linear constraints (*non-lin*) - unless the solver can make simplifications during pre-processing. The number of non-linear constraints (*non-lin*) is especially significant as it is indicative of the amount of computational effort required to produce a tight linear or convex relaxation.

Table 6.2 reports the execution times for COUENNE and BARON. In cases where

	COUENNE					BARON
Problem	*no-bt*	*fbbt*	*obbt*	*afbbt*	*strong*	
camshape100	(-5.2243)	(-5.0462)	(-4.8896)	(-5.0462)	**(-4.6698)**	(-4.7754)
ex3_1_1	3 s	2 s	3 s	2 s	1 s	**< 1 s**
ex3_1_2	< 1 s	< 1 s	< 1 s	**< 1 s**	< 1 s	< 1 s
ex3_1_3	< 1 s	< 1 s	< 1 s	**< 1 s**	< 1 s	< 1 s
ex5_4_4	524 s	24 s	179 s	25 s	(0.0070)	**16 s**
ex7_3_4	*Num*	3 s	909 s	5 s	(3.3125)	**1 s**
ex7_3_5	*Num*	19 s	595 s	32 s	(0.3629)	**1 s**
ex8_4_4	39 s	**< 1 s**	10 s	2 s	2 s	**< 1 s**
ex8_4_7	(2.7630)	*Num*	(27.5899)	*Num*	**576 s**	(23.5802)
hhfair	($-\infty$)	($-\infty$)	($-\infty$)	($-\infty$)	**6 s**	5684 s
meanvar	1 s	**< 1 s**	3 s	**< 1 s**	1 s	**< 1 s**
qp1	($-\infty$)	**(-0.1514)**	($-\infty$)	(-0.1687)	(-0.2440)	(-0.7681)
qp2	($-\infty$)	**(-0.1495)**	($-\infty$)	(-0.1603)	(-0.2478)	(-0.7617)
qp3	($-\infty$)	(-0.3658)	($-\infty$)	(-0.4321)	(-0.2488)	**(-0.1364)**
st_m1	($-\infty$)	**< 1 s**	(-881939)	1 s	**< 1 s**	**< 1 s**
st_m2	($-\infty$)	**< 1 s**	($-\infty$)	1 s	**< 1 s**	**< 1 s**
st_qpk1	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
st_qpk2	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
st_qpk3	**< 1 s**	**< 1 s**	1 s	**< 1 s**	**< 1 s**	3 s

Table 6.2: Execution times for NLP benchmark problems. In cases that fail to converge, the value of $\underline{z}$ is shown instead in brackets.

convergence was not achieved within the 2 hour time limit, the best possible minimum value ($\underline{z}$) is reported instead of the execution time. The cases that encountered numerical issues are denoted by *Num*. The best overall execution times (or the best $\underline{z}$) for each problem are highlighted in bold.

The results demonstrate no clear differences between COUENNE and BARON in terms of performance. Of the eight problems that required more than one second to evaluate by at least one of the solvers [1], BARON was the fastest on four occasions, while COUENNE/{*fbbt, strong*} were the fastest for the rest. COUENNE/*no-bt* was generally the worst performing configuration overall, which is likely due to the absence of any form bounds tightening.

There were only slight differences between the performance profiles of COUENNE/*fbbt* and COUENNE/*afbbt*. COUENNE/*strong*, while performing reasonably well, failed to obtain solutions for ex_5_4_4, ex_7_3_4, and ex_7_3_5, all three

[1] ex3_1_1, ex5_4_4, ex7_3_4, ex7_3_5, ex8_4_4, ex8_4_4, meanvar, st_m1, and st_m2

	Basis	*var*	*aux*	*lin*	*non-lin*
H_2	STO-2G	2	14	2	10
	STO-3G	2	14	2	10
	3-21G	4	56	2	50
	6-31G	4	56	2	50
HeH^+	STO-2G	2	14	2	10
	STO-3G	2	14	2	10
	3-21G	4	56	2	50
	6-31G	4	56	2	50
LiH	STO-2G	6	109	4	99
	STO-3G	6	109	4	99
H_3^+	STO-2G	3	29	2	24
	STO-3G	3	29	2	24
	3-21G	6	171	2	163
	6-31G	6	171	2	163

Table 6.3: Size and complexity of point energy calculation problems.

of which the other configurations, except for COUENNE/*no-bt*, had no trouble in evaluating within time limits. COUENNE/*fbbt* and COUENNE/*afbbt* both encountered numerical issues for ex_8_4_7.

6.4 Point Energy Calculation

Point energy calculations involve the global minimization of the Hartree-Fock energy with respect to the molecular orbitals, while keeping the atomic coordinates fixed. The electronic state that corresponds to the global minimum is known as the *ground state*. The deterministic global optimization of the Hartree-Fock energy for point energy calculation was first explored by Lavor et. al. [79, 80, 86] for small single atom systems such as Helium and Beryllium using minimal basis sets. This section extends their work by considering larger systems involving molecules including H_2, HeH^+, LiH, and H_3^+; with a larger variety of basis sets, including STO-2G, STO-3G, 3-21G, and 6-31G.

Table 6.3 summarises the size and complexity of these problems. The values of the molecular orbital (M.O) coefficients are allowed to vary from -1.0 to 1.0. Notice that compared to the solvable NLP problems considered previously such as ex5_4_4, ex7_3_4, and ex7_3_5, the point energy problems have fewer degrees of freedom

			COUENNE				BARON	GlobSol
	Basis	Energy (h)	*fbbt*	*obbt*	*afbbt*	*strong*		
H_2	STO-2G	-1.0934	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
	STO-3G	-1.1167	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
	3-21G	-1.1230	**67 s**	1,477 s	72 s	397 s	72 s	1,111 s
	6-31G	-1.1268	**84 s**	1,883 s	85 s	487 s	90 s	1,663 s
HeH^+	STO-2G	-2.7523	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
	STO-3G	-2.8404	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
	3-21G	-2.8873	**30 s**	454 s	33 s	134 s	43 s	623 s
	6-31G	-2.9095	**21 s**	453 s	25 s	84 s	24 s	557 s
LiH	STO-2G	-7.5497	**5,512 s**	(-13.3648)	5,568 s	(-7.7085)	(-7.5535)	-
	STO-3G	-7.8046	**5,272 s**	(-12.9523)	5,359 s	(-7.9342)	(-7.8090)	-
H_3^+	STO-2G	-1.2066	**2 s**	20 s	3 s	8 s	2 s	14 s
	STO-3G	-1.2308	2 s	21 s	3 s	7 s	< **1 s**	15 s
	3-21G	-1.2701	**(-8.2592)**	(-82.4512)	(-8.3232)	(-50.4881)	(-9.0035)	-
	6-31G	-1.2743	(-9.4666)	(-87.9969)	**(-9.3316)**	(-45.3051)	(-9.6552)	-

Table 6.4: Execution times for point energy calculation. In cases that fail to converge, the value of $\underline{z}$ is shown instead in brackets.

but greater numbers of non-linear constraints. Since the point energy problem is an instance of polynomial programming, it can be solved using most conventional deterministic MINLP solvers. This provides an opportunity to directly evaluate the performance of COUENNE, BARON, and GlobSol within the Hartree-Fock context. It should be noted that a conventional SCF solver would be able to solve any of the problems considered here in less than one second.

Table 6.4 reports the ground state energy and the execution times for each system. The first two columns specify the molecule and basis set involved, the third column shows the ground state energy or the best feasible energy value (in Hartrees) that is found within the time limits ($\bar{z}$). The remaining columns show the execution times for COUENNE, using *fbbt*, *obbt*, *afbbt*, and *strong*, BARON, and finally GlobSol. In cases where convergence was not achieved within the 2 hour time limit, the best possible energy value ($\underline{z}$) is reported instead except in the GlobSol calculations where we were unable to obtain this information. The best execution time (or the best $\underline{z}$) for each problem is highlighted in bold.

The results show that COUENNE/*fbbt* was the fastest approach overall, followed closely by BARON and COUENNE/*afbbt*. The performance of the different COUENNE configurations varied significantly. COUENNE/*obbt* was generally one or two orders of magnitude slower than the other configurations; which suggests that the absence of feasibility based bounds tightening has a significant detrimental effect on performance. The results also showed that COUENNE/*strong* was not as fast

as COUENNE/*fbbt* or COUENNE/*afbbt*. On the other hand, it was also observed that COUENNE/*strong* required significantly fewer nodes to reach convergence. For example in HeH^+/6-31G, *fbbt* required 5,401 nodes, whereas *strong* required only 1,901 nodes. This suggests that sophisticated heuristics such as strong branching may not be necessary for smaller problems due to its overhead, but may be significant when evaluating larger problems where memory usage becomes an issue.

The execution times for `GlobSol` were generally one or two orders of magnitude slower than the fastest execution times, but was still comparable to those of COUENNE/*obbt*. This is to be expected given that `GlobSol` uses an interval analysis based approach, however the additional time to solution may be a reasonable compromise for the additional reliability that interval-based approaches provide. Furthermore, even though COUENNE/*fbbt* was marginally faster than the default BARON configuration, the performance of BARON can presumably be improved given further tuning. Nevertheless, the results in this section again demonstrates that the performance of COUENNE is comparable to that of BARON.

The results also show that LiH/STO-2G, H_3^+/3-21G, and H_3^+/6-31G were particularly difficult to solve. Applying the profiling tool `gprof` it was found that the majority of the execution time ($> 95\%$) in COUENNE for these and other point energy problems was spent on solving linear relaxations (LP_k). This emphasizes the importance of the LP solver to the performance of a branch and bound solver.

To gain further insight on these performance issues, we also examine the type of nodes that are evaluated in a typical calculation. Figure 6.1 illustrates the cross-section of nodes (N_k) evaluated in HeH^+/3-21G (using COUENNE/*fbbt*). The x-axis represents the range of the first molecular orbital coefficient $c_{1,1}$, and the y-axis represents the range of the second molecular orbital coefficient $c_{1,2}$. The nodes from early iterations are illustrated in a lighter shade, which gradually becomes darker for nodes in later iterations. An ϵ-convergent solution to this problem is located at the point $c_{1,1}$ = 0.4668 and $c_{1,2}$ = 0.5250.

Figure 6.1 illustrates the gradual decrease in the node size as the procedure progresses, due to branching. The vast majority of these nodes appear to concentrate near the solution, and a *symmetric solution* at the point $c_{1,1}$ = -0.4668, and $c_{1,2}$ = -0.5250. This is to be expected given that nodes that are nearer to a solution are generally more difficult to fathom based on non-optimality since it is near the optimal solution, and also difficult to fathom based on infeasibility since it is near a feasible solution. In both cases, the linear relaxation must be extremely tight in order to allow fathoming, which often requires existing nodes to be branched into smaller nodes. The figure also shows that a substantial portion

of the execution time is spent doing redundant work in evaluating nodes near a symmetric solution. Removing symmetric solutions from the formulation of the Hartree-Fock equations may therefore be significant in reducing execution times.

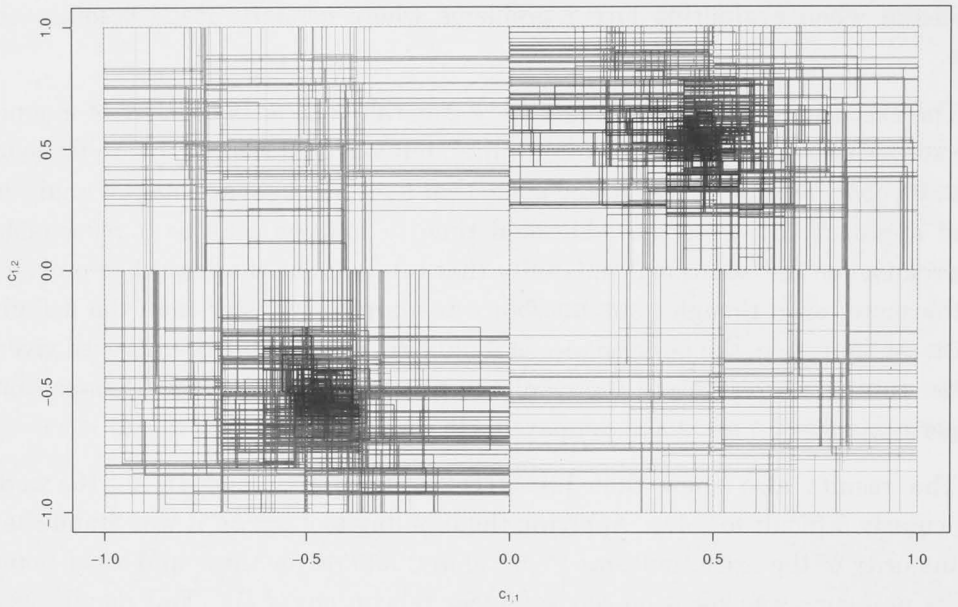


Figure 6.1: A cross-section of nodes evaluated in the point energy calculation of HeH^+/3-21G (using COUENNE/*fbbt*).

6.4.1 Removing Variable and Value Symmetry in RHF

The presence of symmetric solutions in a problem increases the number of redundant regions that the branch and bound procedure must search. In certain cases, these symmetric solutions can be removed through the introduction of *symmetry-breaking constraints* [85]

For the purpose of the following discussion, we represent the optimal molecular orbital coefficients as a matrix $C' \in \mathbb{R}^{M \times N}$,

$$C' = \begin{bmatrix} \mathbf{c_{11}} & \mathbf{c_{12}} & \cdots & \mathbf{c_{1N}} \\ \mathbf{c_{21}} & \mathbf{c_{22}} & \cdots & \mathbf{c_{2N}} \\ \vdots & \vdots & \vdots & \ddots \\ \mathbf{c_{M1}} & \mathbf{c_{M2}} & \cdots & \mathbf{c_{MN}} \end{bmatrix} \tag{6.1}$$

where $M = N_{elec}/2$ is the number of molecular orbitals, and N is the number of

atomic orbitals. Furthermore, we let each row refer to a molecular orbital, and each column refer to an atomic orbital.

In optimization, *variable symmetry* refers to a bijection σ on *variable indices* that preserves optimality [159]. That is, if $\{x_i = v_i \mid i = \{1, 2, ..n\}\}$ is a global minimizer then $\{x_{\sigma(i)} = v_i \mid i = \{1, 2, ..n\}\}$ is also a valid global minimizer.

The anti-symmetry property of wavefunctions [148] allows molecular orbitals, and therefore each row in C', to be exchanged without affecting optimality. This results in variable symmetry in the Hartree-Fock equations with $M!$ possible bijections.

To remove this symmetry, it is usually sufficient to define constraints that enforce value ordering for the elements of at least one column in C', for example $c_{11} \geq c_{21} \geq c_{31} .. \geq c_{M1}$. However, there is a possibility that two or more coefficients within that ordering may be equal, allowing them to be exchanged without affecting optimality. To completely remove this symmetry, we can apply *lex leader constraints* [19] for each adjacent row i and $(i + 1)$ in C'. This can be written as

$$\begin{aligned} & & c_{i1} \geq c_{(i+1)1} \\ (c_{i1} = c_{(i+1)1}) & \rightarrow & c_{i2} \geq c_{(i+1)2} \\ (c_{i1} = c_{(i+1)1} \wedge c_{i2} = c_{(i+1)2}) & \rightarrow & c_{i3} \geq c_{(i+1)3} \\ & .... & \\ (\textstyle\bigwedge_{j=1}^{N-1} c_{ij} = c_{(i+1)j}) & \rightarrow & c_{iN} \geq c_{(i+1)N}. \end{aligned} \tag{6.2}$$

So that when two adjacent elements are identical, an ordering is enforced for the elements of the next column, and so on. This relationship can be expressed in short as $c_{i1}, c_{i2}, .., c_{iN} \leq_{lex} c_{(i+1)1}, c_{(i+1)2}, .., c_{(i+1)N}$.

Variable symmetry may also exist between coefficients in systems involving atoms of the same type, such as H_2, H_3^+, and Li_2. The exact relationship between variables depends on the particular molecule, and the type of basis set used. In systems with only one molecular orbital such as H_2 and H_3^+, it may be possible to assume that the coefficient values centered on one atom are equal to the coefficient values centered on the other.

Value symmetry in optimization refers to the bijection σ on *variable values* that preserves optimality [159]. That is, if $\{x_i = v_i \mid i = \{1, 2, ..n\}\}$ is a global minimizer then $\{x_i = v_{\sigma(i)} \mid i = \{1, 2, ..n\}\}$ is also a valid global minimizer.

It can be deduced from Equation 5.32 and Equation 5.33 that any combination of rows in C' can be multiplied by -1 without affecting optimality, where for instance

$$C'' = \begin{bmatrix} -c_{11} & -c_{12} & \cdots & -c_{1N} \\ c_{21} & c_{22} & \cdots & c_{2N} \\ \vdots & \vdots & \vdots & \ddots \\ c_{M1} & c_{M2} & \cdots & c_{MN} \end{bmatrix} \tag{6.3}$$

is also a global minimizer, alongside C'. This results in value symmetry in the Hartree-Fock equations with 2^M possible bijections.

To remove this symmetry, it is usually sufficient to enforce a particular sign for at least one coefficient in each row. For example, in the earlier point energy calculation experiments each coefficient was bounded between -1.0 and 1.0, it is however permissible to bound one coefficient in each row between -1.0 and 0.0 or between 0.0 and 1.0 without losing the optimal solution. However, this coefficient could possibly be equal to zero at the optimal solution, allowing the sign of the other coefficients to be inverted without affecting optimality. To completely remove symmetry, the following lex leader constraints can be defined for each row i:

$$\begin{aligned} & & c_{i1} \geq 0 \\ (c_{i1} = 0) & \rightarrow & c_{i2} \geq 0 \\ (c_{i1} = 0 \wedge c_{i2} = 0) & \rightarrow & c_{i3} \geq 0 \\ & \ldots. & \\ (\textstyle\bigwedge_{j=1}^{N-1} c_{ij} = 0) & \rightarrow & c_{iN} \geq 0. \end{aligned} \tag{6.4}$$

A more difficult form of value symmetry to remove is caused by dependencies between basis functions. This symmetry implies that it is possible to obtain the optimal solution via different mixing of M.O coefficient values. There is no general approach to remove this symmetry, aside from using strictly orthogonal basis functions, which is not practical.

The following experiment investigates the improvements to the execution time due to the introduction of the variable and value symmetry-breaking constraints outlined above. Note however that lex leader constraints were not included in this experiment since the number of constraints it introduces can be very costly. Instead we apply a simple value ordering between the elements of one of the columns of C, and define all elements in this column to be non-negative. These have the same effect of eliminating variable and value symmetry as the lex leader constraints assuming that the coefficients of interest do not have the same value and are not equal to zero.

Table 6.5 shows the execution times when variable and value symmetry-breaking constraints are applied. The results are presented for both COUENNE and

BARON. The layout of this table is similar to Table 6.4. The rows denoted as STD show the execution times using the standard Hartree-Fock formulation outlined Equation 5.32, whereas the rows denoted IMP show the improved execution times when variable and value symmetry-breaking constraints are applied.

As expected, the results show significant improvements in the execution time when symmetry-breaking constraints are applied. All problems involving the H_2 and H_3^+ molecules can now be solved in less than one second. This is presumably due to the removal of variable symmetry with respect to the coefficients of each hydrogen atom. The improvement in problems involving HeH^+ was around a factor of two, this is due to the removal of value symmetry with respect to the sign of the coefficients, which removes one symmetric solution. Finally, the improvement in problems involving LiH was by a factor of seven to eight, this is due to the removal of variable symmetry with respect to the molecular orbitals and the removal of value symmetry with respect to the sign of the coefficients, which leads to the removal of up to $(2! \times 2^2) - 1 = 7$ symmetric solutions in total.

Previous work has focused on applying the reformulation-linearization techniques (RLT) and other convex relaxation techniques to accelerate Hartree-Fock point energy calculations [86, 17, 166]. However, the above results point to the elimination of symmetry in the Hartree-Fock equations as another critical factor that should be investigated in greater detail in future work.

6.4.2 Summary: Point Energy Calculation

In this section, we evaluate the point energy of H_2, HeH^+, LiH, and H_3^+, using basis sets STO-2G, STO-3G, 3-21G, and 6-31G. The solvers used were COUENNE/{*fbbt*, *obbt*, *afbbt*, *strong*}, BARON, and `GlobSol`. The best performing solver overall was COUENNE/*fbbt*, followed closely by BARON and COUENNE/*afbbt*. While `GlobSol` and COUENNE/*obbt* were generally one or two orders of magnitude slower than the rest. COUENNE/*strong* while not the best performing, required the least number of nodes to reach convergence, which may be advantageous when applied to larger problems.

By inspecting the nodes that were evaluated in a typical calculation, it was observed that a substantial amount of execution time is wasted in evaluating regions that are near a symmetric solution. As a response, additional constraints were imposed on the Hartree-Fock equations in order to remove variable and value symmetry in the original formulation. This included constraints that: i) impose an ordering between the coefficient values of different molecular orbitals, ii) eliminate symmetric M.O coefficients in H_2 and H_3^+, and iii) enforce the sign of at least one coefficient of each molecular orbital. Subsequent experiments confirm that the execution time can be reduced significantly when these symmetry-breaking

	Basis	Energy (h)		COUENNE *fbbt*	COUENNE *obbt*	COUENNE *afbbt*	COUENNE *strong*	BARON
H_2	STO-2G	-1.0934	STD	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
			IMP	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
	STO-3G	-1.1167	STD	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
			IMP	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
	3-21G	-1.1230	STD	**67 s**	1,477 s	72 s	397 s	72 s
			IMP	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
	6-31G	-1.1268	STD	**84 s**	1,883 s	85 s	487 s	90 s
			IMP	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
HeH^+	STO-2G	-2.7523	STD	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
			IMP	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
	STO-3G	-2.8404	STD	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
			IMP	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
	3-21G	-2.8873	STD	**30 s**	454 s	33 s	134 s	43 s
			IMP	**15 s**	275 s	17 s	63 s	19 s
	6-31G	-2.9095	STD	**21 s**	453 s	25 s	84 s	24 s
			IMP	**11 s**	248 s	12 s	41 s	11 s
LiH	STO-2G	-7.5497	STD	**5,512 s**	(-13.3648)	5,568 s	(-7.7085)	(-7.5535)
			IMP	**726 s**	(-7.6000)	755 s	5,866 s	841 s
	STO-3G	-7.8046	STD	**5,272 s**	(-12.9523)	5,359 s	(-7.9342)	(-7.8090)
			IMP	**665 s**	(-7.8626)	798 s	6,099 s	757 s
H_3^+	STO-2G	-1.2066	STD	**2 s**	20 s	3 s	8 s	2 s
			IMP	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
	STO-3G	-1.2308	STD	2 s	21 s	3 s	7 s	**< 1 s**
			IMP	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
	3-21G	-1.2701	STD	**(-8.2592)**	(-82.4512)	(-8.3232)	(-50.4881)	(-9.0035)
			IMP	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s
	6-31G	-1.2743	STD	(-9.4666)	(-87.9969)	**(-9.3316)**	(-45.3051)	(-9.6552)
			IMP	< 1 s	< 1 s	< 1 s	< 1 s	< 1 s

Table 6.5: Execution times for point energy calculation. The rows labelled STD show the execution times for the standard Hartree-Fock formulation, while the rows labelled IMP show the execution times when symmetry-breaking constraints are applied. In cases that fail to converge, the value of $\underline{z}$ is shown instead in brackets.

constraints are applied.

The next section considers the substantially more difficult geometry optimization problems that incorporate non-constant one and two-electron integral terms. As such, BARON and GlobSol can no longer be used. The *afbbt* and *obbt* configurations are factored out, the former due to its similarity to *fbbt*, and the latter due to its relatively poor performance. The symmetry-breaking constraints outlined in this section will also be applied to the geometry optimization formulation.

		A1 - $F_m(T)$				A2 - $(ss\|ss)$			
	Basis	*var*	*aux*	*lin*	*non-lin*	*var*	*aux*	*lin*	*non-lin*
$\mathbf{H_2}$	STO-2G	2	124	49	73	2	53	19	32
	STO-3G	2	393	151	240	2	86	33	51
	3-21G	3	455	174	278	3	180	54	123
	6-31G	3	1,040	393	644	3	227	73	151
$\mathbf{HeH^+}$	STO-2G	3	209	79	127	3	86	29	54
	STO-3G	3	731	279	449	3	148	55	90
	3-21G	5	843	311	527	5	286	80	201
	6-31G	5	1,980	743	1,232	5	372	116	251
LiH	STO-2G	7	840	275	558	7	322	74	241
$\mathbf{H_3^+}$	STO-2G	6	793	310	480				
	STO-3G	6	3,091	1,188	1,900				

Table 6.6: Size and complexity of geometry optimization problems.

6.5 Geometry Optimization

We consider the problem of simultaneously finding the Hartree-Fock wavefunction and the lowest-energy equilibrium geometry of the H_2^+, HeH^+, LiH, and H_3^+ molecules using the STO-2G, STO-3G, 3-21G, and 6-31G basis sets. These problems also allow us to evaluate the performance of the $F_m(T)$-based (from Section 5.4.2) and the $(ss|ss)$-based (from Section 5.4.3) linear relaxation approaches - and also to compare the different bounds tightening and branch selection algorithms.

The size and complexity of each problem is briefly summarized in Table 6.6 where *var* denotes the number of decision variables, and *lin* and *non-lin* denote, respectively, the number of linear and non-linear constraints in the standardized formulation.

The complexity of the problem formulation with respect to the $F_m(T)$ decomposition approach (denoted by A1 - $F_m(T)$) is several times greater than that of the $(ss|ss)$ decomposition approach (denoted by A2 - $(ss|ss)$) due to the number of $F_m(T)$ expressions created. The largest problem in terms of the number of linear and non-linear constraints is H_3^+/STO-3G. This has 6 decision variables, 1,188 linear constraints and 1,900 non-linear constraints for A1 - $F_m(T)$. The largest diatomic problem is HeH^+/6-31G. This has 5 decision variables, 743 linear constraints and 1,232 non-linear constraints for A1 - $F_m(T)$, and 116 linear constraints and 251 non-linear constraints for A2 - $(ss|ss)$. Finally, the largest problem in terms of

			A1 - $F_m(T)$		A2 - $(ss\|ss)$	
	Basis	Energy (h)	*fbbt*	*strong*	*fbbt*	*strong*
$\mathbf{H_2}$	STO-2G	-1.0938	**1 s**	9 s	2 s	11 s
	STO-3G	-1.1175	**5 s**	310 s	11 s	50 s
	3-21G	-1.1230	1,837 s	796 s	**445 s**	800 s
	6-31G	-1.1268	1,919 s	2,494 s	**1,045 s**	2,133 s
$\mathbf{HeH^+}$	STO-2G	-2.7653	306 s	174 s	111 s	**103 s**
	STO-3G	-2.8544	1,436 s	1,591 s	**381 s**	487 s
	3-21G	-2.8874	*Mem*	(-2.9968)	(-14.0850)	**88,305 s**
	6-31G	-2.9099	*Mem*	(-6.2056)	(-24.9989)	**(-3.1112)**
LiH	STO-2G	-7.5494	*Mem*	(-11.4437)	(-15.2161)	**(-9.5161)**
$\mathbf{H_3^+}$	STO-2G	-1.2248	*Mem*	**4,746 s**		
	STO-3G	-1.2469	*Mem*	**69,051 s**		

Table 6.7: Execution times for geometry optimization. In cases that fail to converge, the value of $\underline{z}$ is shown instead in brackets.

the number of decision variables is LiH/STO-2G, which has 7 decision variables in total.

In diatomic molecules such as H_2, HeH^+, and LiH, the geometry is optimized with respect to the *bond distance* between the two atoms. The bond distances for H_2 and HeH^+ are bounded between 1.0 R_e and 3.0 R_e (Bohr radius), while for LiH between 2.0 R_e and 4.0 R_e. The geometry of H_3^+ is constrained to be the shape of an isosceles triangle $\triangle ABC$, where the line segments $\overline{AB}$ and $\overline{AC}$ are equidistant. The optimization is performed with respect to the *bond angle* $\angle BAC$, and the *bond distance* $\overline{AB}$. The bond angle is bounded between $45°$ and $90°$ and the bond distance between 1.0 R_e and 3.0 R_e. The molecular orbital coefficients in each system are allowed to vary between -1.0 and 1.0, but with the addition of symmetry-breaking constraints - as was previously applied to the point energy problems in Section 6.4.1.

Table 6.7 reports the results from the geometry optimization experiment. The table is organized as follows. The first two columns specify the molecule and basis set involved, the third column shows the ground state energy or the best feasible energy value that is found within the time limits ($\bar{z}$). The cases that are halted due to lack of memory are denoted by *Mem*. The fourth and fifth columns show the execution time (in seconds) required for the A1-$F_m(T)$ approach using *fbbt* and *strong*, respectively. Finally, the sixth and seventh columns show the execution time (in seconds) required for the A2-$(ss|ss)$ approach using *fbbt* and

				A1 - $F_m(T)$		A2 - $(ss\|ss)$		
	Basis	Energy (h)	Ang°, R_e	Time (z)	Best	Time (z)	Best	Ratio
H_2	STO-2G	-1.0938	1.3659	1 s	(*fbbt*)	2 s	(*fbbt*)	1:(**2.00**)
	STO-3G	-1.1175	1.3404	5 s	(*fbbt*)	11 s	(*fbbt*)	1:(**2.20**)
	3-21G	-1.1230	1.3908	796 s	(*strong*)	445 s	(*fbbt*)	(**1.79**):1
	6-31G	-1.1268	1.3847	1,919 s	(*fbbt*)	1,045 s	(*fbbt*)	1:(**1.84**)
HeH^+	STO-2G	-2.7653	1.7136	174 s	(*strong*)	103 s	(*fbbt*)	(**1.69**):1
	STO-3G	-2.8544	1.7512	1,436 s	(*fbbt*)	381 s	(*fbbt*)	(**3.77**):1
	3-21G	-2.8874	1.4782	(-2.9968)	(*strong*)	88,305 s	(*strong*)	-
	6-31G	-2.9099	1.4498	(-6.2056)	(*strong*)	(-3.1112)	(*strong*)	-
LiH	STO-2G	-7.5494	2.9767	(-11.4437)	(*strong*)	(-9.5161)	(*strong*)	-
H_3^+	STO-2G	-1.2248	59.91°, 1.8381	4,746 s	(*strong*)			-
	STO-3G	-1.2469	59.49°, 1.8298	69,051 s	(*strong*)			-

Table 6.8: Summary of geometry optimization results.

strong, respectively. If convergence was not achieved, the value of the best lower bound $\underline{z}$ is reported instead within parentheses. The fastest execution time (or the best $\underline{z}$) for each problem is highlighted in bold.

The equilibrium geometry was found to within the 1KCal/Mol threshold for all but two problems within the stated time limit of 36 hours. Out of the successful cases, HeH^+/3-21G required the greatest amount of time at 88,305 s for the A2-$(ss|ss)$ using *strong*. A1-$F_m(T)$ failed to converge for the same problem, reaching a lower bound of -2.9968 at the time limit. Both A1-$F_m(T)$ and A2-$(ss|ss)$ were unable to find the optimal solution for HeH/6-31G and LiH/STO-2G. Nevertheless, A2-$(ss|ss)$ was able to achieve the better lower bound for both problems.

The results of Table 6.7 are further summarized in Table 6.8. The third column shows the optimal bond distance in Bohr radius in the diatomic cases, and the optimal bond angle and bond distance in H_3^+. The fastest execution time (or the best lower bound) using A1-$F_m(T)$ and A2-$(ss|ss)$ are given in the fourth and fifth columns, the fastest bounds tightening and branch selection configuration is indicated in parentheses. Finally the seventh column shows the ratio of execution times between A1-$F_m(T)$ and A2-$(ss|ss)$, with the value corresponding to the least efficient approach highlighted in bold font. Note that in cases such as HeH^+/6-31G and LiH/STO-2G, where ϵ-convergence was not achieved, the energy and the bond length reported correspond to that of the best feasible solution found.

In reference to Table 6.8, it should be noted that the bond lengths and bond angles obtained via different solver configurations can vary to some degree, but still yield energy values that are consistent within the ϵ threshold.

6.5.1 Performance Analysis

The results in Table 6.7 and Table 6.8 showed that execution times varied significantly depending on the choice between linear relaxation schemes A1-$F_m(T)$ and A2-$(ss|ss)$, and the choice between bounds tightening and branch selection configurations *fbbt* and *strong*. The following section investigates the factors influencing these differences, and explores the overall performance characteristics of the solver.

6.5.1.1 Number of Linear Constraints and Nodes Generated

The differences between A1-$F_m(T)$ and A2-$(ss|ss)$, and the differences between *strong* and *fbbt*, in terms of the number of linear constraints generated, and the number of nodes evaluated are shown in Figure 6.2 and Figure 6.3, respectively.

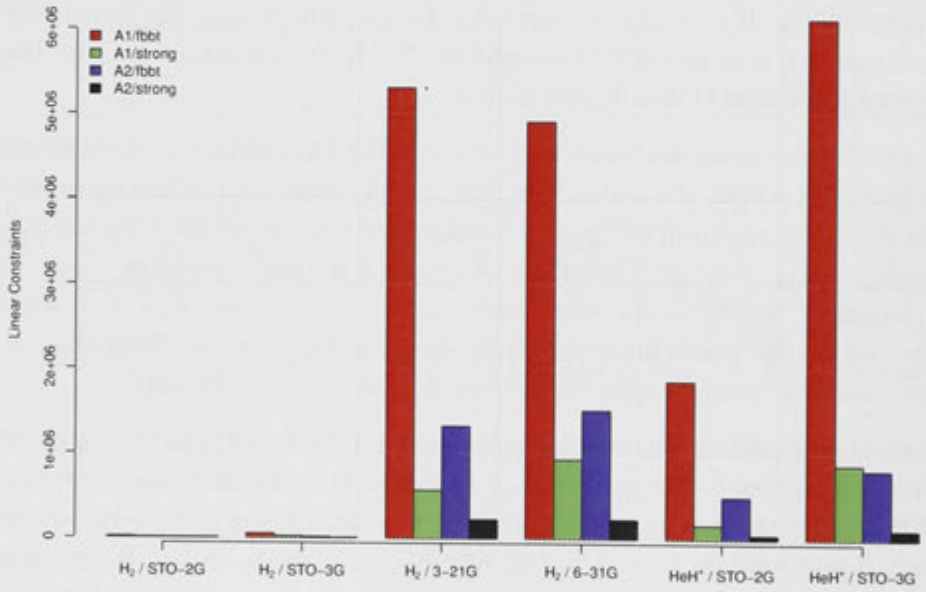


Figure 6.2: The number of linear constraints generated prior to ϵ-convergence.

The results in Table 6.7 shows that the A1-$F_m(T)$ approach is generally more efficient for smaller problems sizes such as H_2/STO-2G and H_2/STO-3G, while the A2-$(ss|ss)$ approach is generally more efficient for larger problem sizes such as H_2/3-21G, H_2/6-31G, HeH^+/STO-2G, HeH^+/STO-3G, HeH^+/3-21G, and HeH^+/6-31G.

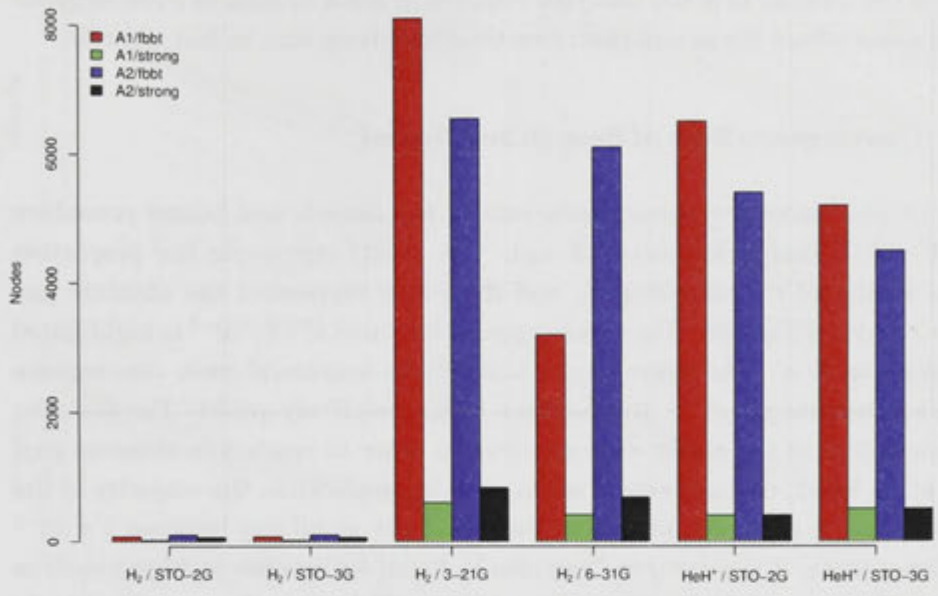


Figure 6.3: The number of nodes evaluated prior to ϵ-convergence.

Even though the number of linear and non-linear constraints associated with A1-$F_m(T)$ is far greater than A2-$(ss|ss)$, the additional complexity that this entails does not fully explain the differences in performance. On the contrary, using more linear constraints may result in tighter lower bounds and therefore fewer nodes to evaluate (as shown in Figure 6.3). A more likely explanation for the cases where A1-$F_m(T)$ performed worse than A2-$(ss|ss)$ is the strain on the memory subsystems that results from the sheer number of the linear constraints generated in A1-$F_m(T)$, and the additional computation required to evaluate them. Giving rise to the clear differences between the number of linear constraints generated for A1-$F_m(T)$ and A2-$(ss|ss)$ shown in Figure 6.2.

The results also show that *fbbt* was generally the best configuration for solving smaller systems, while *strong* was the best at solving larger systems. Larger systems such as HeH^+/3-21G and HeH^+/6-31G were not solvable using A1-$F_m(T)$/*fbbt* because the test machine lacks sufficient memory. This is directly due to the large number of linear constraints generated by A1-$F_m(T)$ combined with the large number of nodes generally required for *fbbt* to converge.

The more efficient use of nodes by *strong* is an important factor in larger systems, where greater amounts of time are required to evaluate each node. In contrast, *fbbt* with less overhead can use "brute force" to search for the solution

in smaller systems. Figure 6.2 shows that *strong* generally requires only a small fraction of the number of nodes that *fbbt* requires in order to achieve ϵ-convergence - even in cases where the actual execution time for *strong* was, in fact, greater.

6.5.1.2 Convergence Rate of Branch and Bound

Figure 6.4 illustrates the convergence rate of the branch and bound procedure for HeH^+/3-21G (using A2-$(ss|ss)$/*strong*). The x-axis represents the proportion of nodes evaluated (in percentages), and the y-axis represents the absolute gap between $\overline{z}$ and $\underline{z}$ in log scale. The ϵ-convergence threshold of 1.6×10^{-3} is highlighted by a horizontal line. The figure shows that, in the successful case, convergence towards an absolute gap of 1×10^{-1} Hartrees occurs relatively quickly. For example, only around 50% of the nodes were required in order to reach this absolute gap. On the other hand, the majority of nodes, and by implication the majority of the execution time, is expended in closing the seemingly small gap between 1×10^{-1} and ϵ-convergence. A similar trend can also be found for smaller problems such as for H_2/3-21G, H_2/6-31G, HeH^+/STO-2G, and HeH^+/STO-3G as illustrated in Figure 6.5.

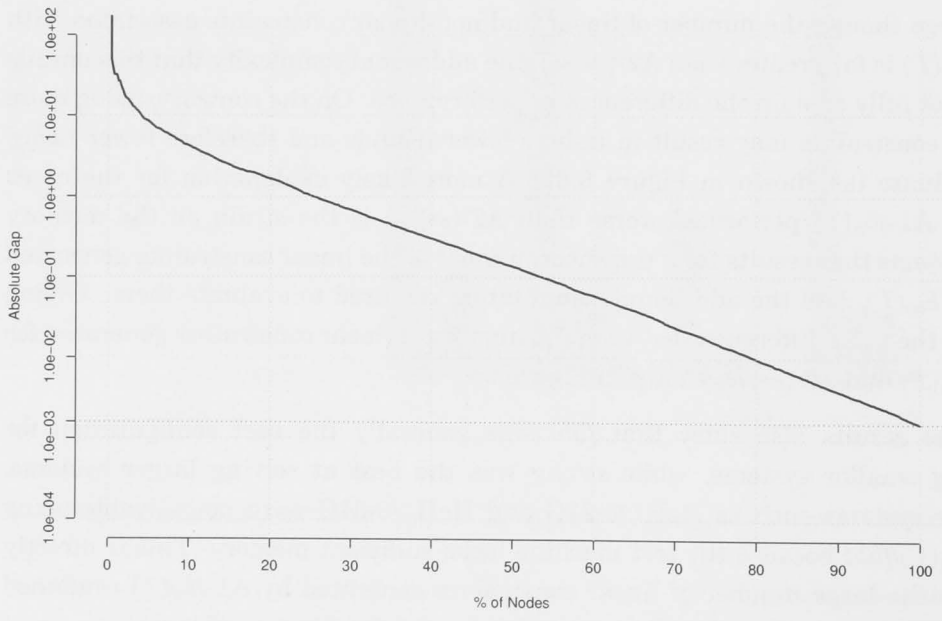


Figure 6.4: The absolute gap between $\underline{z}$ and $\overline{z}$ as a function of the number of nodes evaluated (expressed in %) for HeH^+/3-21G using A2-$(ss|ss)$/*strong*.

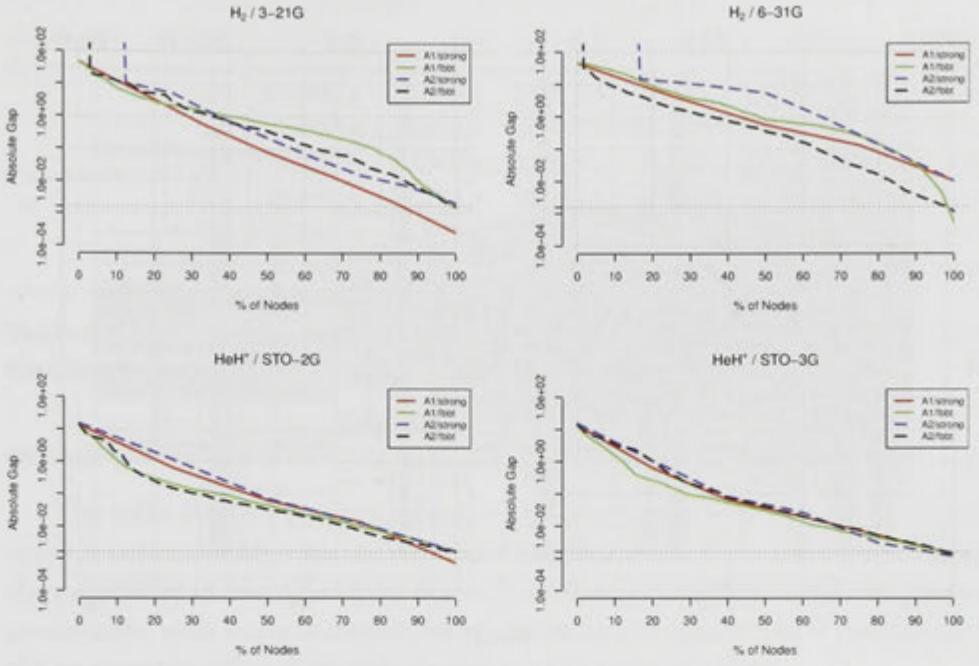


Figure 6.5: The absolute gap between $\underline{z}$ and $\overline{z}$ as a function of the number of nodes evaluated (expressed in %) for H_2/3-21G, H_2/6-31G, HeH^+/STO-2G, and HeH^+/STO-3G.

6.5.1.3 Composition of Nodes Evaluated

Figure 6.6 illustrates a cross-section of all nodes (N_k) evaluated in the geometry optimization of HeH^+/3-21G (using A2-$(ss|ss)$/*strong*). The x-axis represents the range of bond distances, while the y-axis represents the range of the first molecular orbital coefficient $c_{1,1}$. The nodes from early iterations are illustrated in a lighter shade, which becomes increasingly dark for nodes in later iterations. An ϵ-convergent solution to this problem is located at the point $(1.4782, 0.4647)$.

Like the cross-section for point energy calculation in Figure 6.1, this figure also shows that the majority of the nodes are concentrated near the solution. These nodes are difficult to fathom based on non-optimality or infeasibility unless they are further partitioned into smaller nodes. This result is in agreement with earlier observations that a large portion of the execution time is expended in closing a relatively small gap between 1×10^{-1} and ϵ-convergence.

One way that this clustering problem can be avoided is to add constraints which excludes the region near the best solution found, however it is possible that one of the excluded points may contain a better solution or that the constraints may cause ill-conditioning.

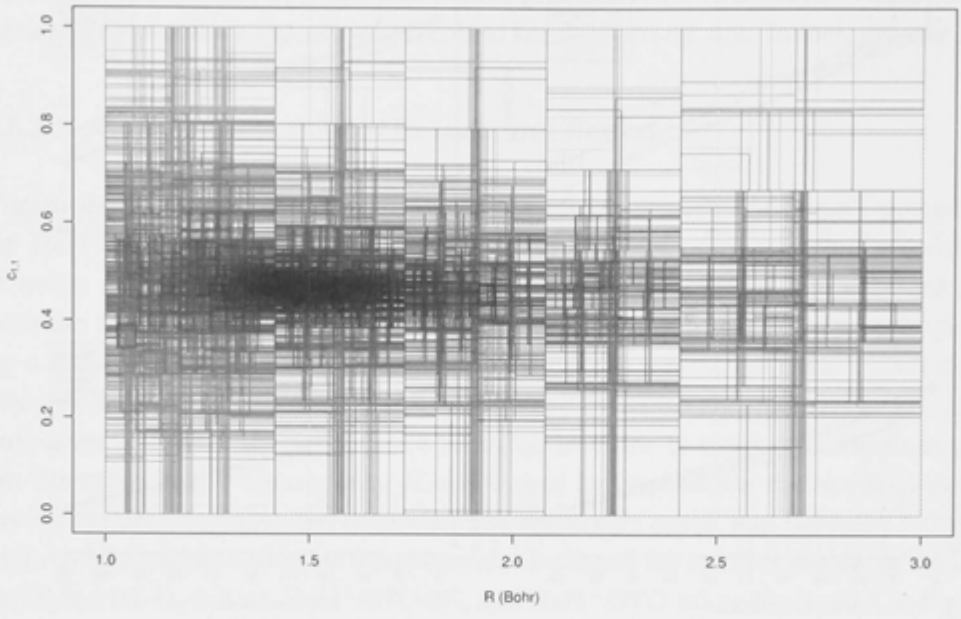


Figure 6.6: A cross-section of nodes evaluated in the geometry optimization of HeH^+/3-21G (using A2-$(ss|ss)$/*strong*).

6.5.1.4 Effect of Varying Variable Bound Constraints

The variable bound constraints used so far are relatively wide. That is, we do not make any specific assumptions about the system being considered. However, in actual applications it may be possible to incorporate a degree of prior knowledge in order to reduce the search space, for example by varying the constraints or the variable bound constraints. Motivated by this observation, the following experiment investigates the effect of changing the variable bound constraints on the execution time for HeH+ / 321G (using the A2-$(ss|ss)$ approach).

Table 6.9 shows the execution times when the bond distances are constrained within an interval width (r_A-*wid*) of 1.0, 0.5, 0.25, 0.125, and 0.0625 R_e around the actual solution of 1.4782 R_e; and if each M.O coefficient value is constrained to within an interval width ($\{c_{\mu a}\}$-*wid*) of 1.0, 0.5, 0.25, 0.125, and 0.0625 around the actual solution of $[0.4647, 0.5309, 0.2024, 0.0190]$. It should be noted that the bond distance in the original problem was constrained between 1.0 R_e and 3.0 R_e giving an absolute width of r_A-*wid* = 2.0 while the coefficient values were (generally) constrained between -1.0 and 1.0 giving an absolute width of $\{c_{\mu a}\}$-*wid* = 2.0. The original geometry optimization problem required 88,305 seconds to converge (see Table 6.7), while its point energy equivalent (where r_A-*wid* = 0.0) only required 63

HeH^+ /3-21G		r_A-*wid* = 2.0	1.0	0.5	0.25	0.125	0.0625	0.0
$\{c_{\mu a}\}$-*wid*=	2.0	**88,305 s**	79,989 s	54,617 s	36,347 s	20,393 s	11,307 s	**63 s**
	1.0	35,645 s	33,533 s	29,420 s	22,743 s	15,157 s	7,107 s	-
	0.5	36,660 s	30,595 s	31,394 s	21,096 s	16,152 s	9,923 s	-
	0.25	26,982 s	23,721 s	22,631 s	18,035 s	12,575 s	6,896 s	-
	0.125	17,661 s	20,635 s	17,395 s	14,665 s	10,475 s	7,039 s	-
	0.0625	10,995 s	10,157 s	9,975 s	9,463 s	7,047 s	4,463 s	-

Table 6.9: Execution times for the geometry optimization of HeH^+/3-21G with differently sized bond distance (r_A-*wid* =) and M.O coefficient ($\{c_{\mu a}\}$-*wid*=) bound constraints.

seconds (see Table 6.5).

The table shows a general reduction in execution time as the size of the search space is reduced with respect to the bond length and the M.O coefficients. However, the reductions in execution time due to variations in the M.O coefficients were less predictable, with some instances where reducing the space of M.O coefficients had the unexpected effect of slightly increasing the execution time.

The results show that even when the search space is reduced to a small, r_A-*wid* = 0.0625 $\times$ $\{c_{\mu a}\}$-*wid* = 0.0625, region, the execution time was still 4,463 seconds. In this case, the (initial) absolute gap between $\underline{z}$ and $\overline{z}$ at the root node was only 0.0562 Hartrees. This is potentially due to the expense involved in fathoming nodes near the ϵ-convergence threshold.

6.5.2 Summary: Geometry Optimization

In this section, we demonstrated the geometry optimization of H_2^+, HeH^+, LiH, and H_3^+ molecules using STO-2G, STO-3G, 3-21G, and 6-31G basis sets. We also analyzed the performance of COUENNE due to factors such as the choice of bounds tightening and branch selection algorithm, the choice of linear relaxation scheme, and the varying of variable bound constraints.

The results show that in smaller problems COUENNE performed better when A1-$F_m(T)$ relaxation and *fbbt* are applied, whereas in larger problems COUENNE performed better when A2-$(ss|ss)$ and *strong* are used. This result can be attributed to factors including i) the overhead in applying the various linearization and branch selection algorithm, ii) the number of linear constraints generated in A1-$F_m(T)$ compared to A2-$(ss|ss)$, and iii) the more efficient use of nodes by *strong* in comparison to *fbbt*.

The results also show that convergence towards an absolute gap of 1×10^{-1}

Hartrees occurs relatively quickly, while the remaining time is expended closing the gap between 1×10^{-1} and the ϵ-convergence threshold. This suggests that more attention should be given to improving the process of eliminating nodes that are near the solution. This issue was further highlighted when we examined the composition of nodes evaluated in a HeH^+/3-21G calculation.

Finally, the effect of changing the variable bound constraints with respect to the bond length and the M.O coefficients was investigated. The general trend demonstrates a reduction in execution time as the search space is reduced, with some exceptions with respect to variations in the space of M.O coefficients.

6.6 Basis Set Optimization

The accuracy of an electronic structure calculation is highly dependent on the choice of basis set. In principle, any desired level of accuracy up to the Hartree-Fock limit can be achieved using a basis set with sufficiently many basis functions. However, since a modest increase in the basis set size can cause an enormous increase in computational cost, a good basis set is one that achieves the desired level of accuracy with as few basis functions as possible. The problem of defining such a basis set is known as *Basis Set Optimization*.

As an illustrative example, Huzinaga [51] defined the optimal four function expansion for Hydrogen to be

$$\begin{aligned} \Phi_1(\mathbf{r}) = \; & 0.50907 g_{1s}(0.123317, \mathbf{r}) + 0.47449 g_{1s}(0.453757, \mathbf{r}) \\ & + 0.13424 g_{1s}(2.04660, \mathbf{r}) + 0.01906 g_{1s}(13.3615, \mathbf{r}) \end{aligned} \tag{6.5}$$

where $g_{1s}(\alpha_{a,i}, \mathbf{r})$ represents a primitive s-Gaussian basis function with exponent $\alpha_{a,i}$, centered at the atomic coordinate $\mathbf{r}$.

A basis set can be constructed from Gaussian basis functions that resemble, as closely as possible, more accurate atomic orbital representations such as the Slater-type orbitals [51]; the parameters of which can be obtained using a least squares fitting. This approach forms the basis for the well known STO-nG type minimal basis sets, e.g. STO-2G, STO-3G, STO-6G.

A more commonly used approach is to construct the basis set from a set of primitive Gaussian functions which minimizes the Hartree-Fock energy [20, 148]. At the Hartree-Fock level, this approach is equivalent to minimizing the Hartree-Fock energy with respect to the set of molecular orbital coefficients $\{c_{\mu,a}\}$, the set of primitive Gaussian exponents $\{\alpha_{a,i}\}$, and the atomic coordinates $\{r_A\}_{A=1}^{N_{atom}}$.

Basis set optimization is a more difficult than either point energy calculation or geometry optimization. In practice, additional constraints are usually imposed to reduce the degree of freedom prior to performing the basis optimization. This often involves defining an approximate parametric relation between Gaussian exponent values, so that the optimization is performed on a limited number of independent parameters instead of the entire set of Gaussian exponents. See [125, 10, 54] for examples.

Basis set optimization has traditionally been the task of quasi-Newton methods such as the *Newton-Raphson* method [23], which like all other local optimization approaches are not guaranteed to converge towards the global minimum. Deterministic global optimization can be used as a more reliable alternative that provides basis functions which are guaranteed to be optimal. Since the $F_m(T)$ decomposition approach can be used to generate linear relaxations for integrals with variable Gaussian exponents, no additional work is required for this task.

The following provides two proof of concept examples for He and H_2, respectively.

Example 6.1. The ground state energy of Helium with respect to two uncontracted s-type Gaussian functions is optimized. The exponents corresponding to each function are allowed to vary between 0 and 1, and 0 and 5, respectively. The molecular orbital coefficients are are bounded between 0 and 1.

The optimal energy was found to be -2.7477 Hartrees, while the corresponding optimal two function basis expansion for Helium is written as

$$\Phi_1(\mathbf{r}) = 0.82102 g_{1s}(0.52825, \mathbf{r}) + 0.28820 g_{1s}(4.01672, \mathbf{r}). \tag{6.6}$$

It is also possible to optimize the basis set and the atomic coordinates simultaneously.

Example 6.2. The ground state energy of H_2 with respect to a single s-type Gaussian function is optimized. The bond distance is bounded between 1 R_e and 2 R_e while the exponent is bounded between 0 and 5. The molecular orbital coefficient is allowed to vary between 0 and 1.

The lowest energy was found to be -0.9808 Hartrees, corresponding to a bond length of 1.5530 R_e. The optimal basis function is written as

$$\Phi_1(\mathbf{r}) = 0.55228 g_{1s}(0.37095, \mathbf{r}). \tag{6.7}$$

Although we only perform basis set optimization on small model systems, the results highlight a promising application area where the high computational

cost of deterministic global optimization can be justified by the potential to generate highly reusable results (such as basis sets). Further details of how the optimized M.O coefficients and Gaussian exponents are combined to form contracted Gaussian functions are discussed in [20] and [148].

6.7 Excited State Calculations

So far we have considered problems that involved locating globally minimal energy states of atoms and molecules. Another area of interest is the locating of higher energy stable states known as *Excited States*. These can be used, for example, to predict electronic spectra [29].

The computational treatment of excited states is challenging given that existing approaches such as the Self-Consistent Field (SCF) method are designed to find stationary points but not any one in particular. At the Hartree-Fock level, excited state calculations are usually performed by introducing specific biases to the Hartree-Fock Hamiltonian. The aim is to either i) exclude the possibility of convergence towards the ground state, or ii) preserve convergence towards a higher energy state; examples of this can be found in [147, 52, 53] as cited by [37]. These biases often have no specific physical or mathematical foundations, and may result in excited states that are not solutions to the original Hartree-Fock equation [37].

While it is possible to find an excited state that is consistent with the Hartree-Fock equations, this usually requires performing the SCF procedure with an initial guess close enough to the desired excited state. This is often more difficult than finding an initial guess for the ground state, since there is less prior information about excited states to base the initial guess upon [37]. That said, a crude initial guess can be formulated by taking the ground state wavefunction and then promoting electrons to virtual orbitals [37].

Deterministic global optimization can be used, not only to find a particular excited state, but to find all the excited states corresponding to a particular molecular system and basis set. At the Hartree-Fock level, an excited state corresponds to a local minimum of the Hartree-Fock energy E_{HF}. All such minima can be found by solving the *First-order Karush-Kuhn-Tucker* (KKT) conditions with respect to the Hartree-Fock equations (RHF). This problem can be expressed in short as the following system of non-linear equations

$$\begin{aligned} KKT: \quad L'_c(\mathbf{c}, \lambda) &= 0 \\ \sum_{a=1}^{N} \sum_{b=1}^{N} c_{\mu,a} c_{\upsilon,b} (\chi_a | \chi_b) &= \delta_{\mu\upsilon} \quad \forall \mu, \upsilon = \{1, 2, 3, .., N_{elec}/2\} \end{aligned} \tag{6.8}$$

where $L(\mathbf{c}, \lambda)$ denotes the Lagrangian dual of the Hartree-Fock equations (Equation 5.32); $\mathbf{c} \in \mathbb{R}^{N_{elec}/2 \times N}$ denotes the set of molecular orbital coefficients; and $\lambda \in \mathbb{R}^{N_{elec}/2}$ denotes the corresponding set of Lagrange multipliers. Note that a solution to this problem is not necessarily an excited state, but could instead be the ground state or a saddle point.

KKT can be expressed as a global optimization problem by the introduction of a slack variable $s \in \mathbb{R}$

$$
\begin{array}{rrcll}
EP: & \min & s & & \\
& L'_c(\mathbf{c}, \lambda) - s & \leq & 0 & \\
& -L'_c(\mathbf{c}, \lambda) + s & \leq & 0 & \\
& \sum_{a=1}^{N} \sum_{b=1}^{N} c_{\mu,a} c_{\mu,b} (\chi_a | \chi_b) - 1 - s & \leq & 0 & \forall \mu = \upsilon \\
& -\sum_{a=1}^{N} \sum_{b=1}^{N} c_{\mu,a} c_{\mu,b} (\chi_a | \chi_b) + 1 + s & \leq & 0 & \forall \mu = \upsilon \\
& \sum_{a=1}^{N} \sum_{b=1}^{N} c_{\mu,a} c_{\upsilon,b} (\chi_a | \chi_b) - s & \leq & 0 & \forall \mu \neq \upsilon \\
& -\sum_{a=1}^{N} \sum_{b=1}^{N} c_{\mu,a} c_{\upsilon,b} (\chi_a | \chi_b) + s & \leq & 0 & \forall \mu \neq \upsilon \\
& s & \geq & 0 &
\end{array}
\tag{6.9}
$$

so that the objective $s = 0$ if and only if the minimizer satisfies the first-order KKT conditions with respect to the Hartree-Fock equations. To find all the excited states, we must therefore find all global minimizers of EP where $s = 0$. This can be achieved using the modified branch and bound approach outlined in Section 5.2.5.

To demonstrate the application of this approach, we consider the problem of finding the excited states of He, Be^{2+}, and Be atoms. For this, we use extended basis sets including 3-21G, 6-31G, and cc-pVDZ, which have virtual orbitals capable of describing excited electronic states. The molecular orbital coefficients are allowed to vary between -1.5 and 1.5, and the Lagrange multipliers are allowed to assume any real value. The results of these calculations are summarized in Table 6.10: the first two columns show the molecule and basis set; the third column shows the energy values; the fourth column shows the corresponding electronic configuration expressed as an M.O coefficient matrix; and finally the fifth column describes the type of energy state being depicted.

In summary, we showed that all the excited states corresponding to a particular system and basis set can be found using deterministic global optimization. This is achieved by expressing the first-order KKT conditions in relation to the system as an optimization problem. This problem consists of multiple global minimizers, each corresponding to a stable energy state. The modified branch and bound scheme outlined in Section 5.2.5 is used to find all such global minimum.

	Basis	Energy (h)	M.O Coeff	State
He	3-21G	-2.8357	[0.4579 0.6573]	Ground
		1.7583	[1.1633 −1.0477]	Excited
	6-31G	-2.8552	[1.0000 0.5136]	Ground
	cc-pVDZ	-2.8552	[0.5921 0.5136 0.0000 0.0000 0.0000]	Ground
		0.5945	[1.1212 −1.2094 0.0000 0.0000 0.0000]	Excited
		2.0034	[0.0000 0.0000 0.4452 0.8954 −0.0075]	Excited
Be^{2+}	3-21G	-13.5303	[0.9931 0.0734 0.0274]	Ground
		-3.1262	[0.1632 −0.8503 −0.1936]	Excited
	6-31G	-13.6097	[0.9983 0.0134 −0.0044]	Ground
		-3.1687	[0.1609 −1.0676 0.0577]	Excited
Be	3-21G	-14.4868	$\begin{bmatrix} -0.8364 & -0.1578 & -0.2858 \\ 0.5767 & -0.1831 & -0.7716 \end{bmatrix}$	Ground
		-13.2952	$\begin{bmatrix} -0.3946 & 1.4836 & -1.3417 \\ 0.9112 & 0.7063 & -0.5954 \end{bmatrix}$	Excited
	6-31G†	-14.5668	$\begin{bmatrix} 0.2226 & -0.2888 & -0.7610 \\ -0.9980 & -0.0161 & 0.0055 \end{bmatrix}$	Ground
		-13.5811	$\begin{bmatrix} 0.6442 & -1.5000 & 1.4547 \\ 0.7625 & 1.3249 & -1.2719 \end{bmatrix}$	Excited

Table 6.10: HF/3-21G, HF/6-31G, [and HF/cc-pVDZ] ground and excited state configurations of He, Be^{2+}, and Be Atoms.

6.8 Conclusion and Future Work

This chapter demonstrates the application of deterministic global optimization in Hartree-Fock theory. Proof of concept calculations were performed in application areas including i) point energy calculation, ii) geometry optimization, iii) basis set optimization, and iv) excited state calculation.

The chapter begins by presenting a preliminary study evaluating the performance of COUENNE and BARON in solving a set of benchmark NLP problems. The results demonstrates no clear differences between the solvers. Given that BARON is known to be very efficient, this result infers that COUENNE is an appropriate option for solving problems of this class.

Point energy calculations were then performed using COUENNE, BARON, and

GlobSol. The results showed that COUENNE and BARON provided the best performance overall. In COUENNE, the inclusion of feasibility-based bounds tightening appears to be a particularly important factor in improving performance, whereas the use of strong branching only bought additional overheads. Although the performance of GlobSol was still reasonable compared to COUENNE and BARON, it may not be a viable option as larger problems are considered. This experiment also demonstrated that the execution time can be improved significantly through the introduction of symmetry-breaking constraints.

The chapter then considers the substantially more complicated geometry optimization problem, which can only be solved using the approaches developed in this thesis. The results show that the most efficient combination of approaches for solving the larger problems are those based on i) the $(ss|ss)$ integral linear relaxation scheme, and ii) strong branching-based heuristics - the former because it uses fewer linear relaxations than the $F_m(T)$ decomposition approach, the latter because it makes better branching decisions and thus processes to reach convergence. The findings also highlight that a disproportionate part of the execution time is spent evaluating nodes near the ϵ-convergence threshold. This is a significant factor to be addressed in future work.

The geometry optimization results also suggest that the effort put towards the development of the $(ss|ss)$ integral relaxation approach in Section 5.4.3 were justified; and suggests that future work in developing relaxations for larger composite expressions within the Hartree-Fock equations may yield further improvements.

The chapter then concludes by discussing the applicability of deterministic global optimization to basis set optimization and excited state calculation; proof of concept calculations were demonstrated for both areas.

The results of this chapter showed that the Hartree-Fock equations can be solved accurately, within a guaranteed margin of error, using deterministic global optimization techniques. However, the results also point to practical computational difficulties that must be overcomed for this approach to be useful in systems larger than those considered so far. Inherent to this is the worst-case exponential time property of deterministic search, and the rapid increase in the size of the primal Hartree-Fock problem even in relatively small systems. Symmetry and numerical instability related issues may also become more acute as larger problems, with larger numbers of terms and more degrees of freedom, are considered.

Part

IV

Concluding Remarks

Chapter 7

Conclusion and Future Work

The enormous computational challenges inherent to electronic structure theory continues to motivate the development of ever more efficient numerical methods. Amidst the successes reported in this endeavour, there remain underlying issues relating to the quality of results obtained by these methods. We observed that these quality issues often stem from two factors. These include i) numerical errors due to the use of finite precision arithmetic and the application of numerical approximations, and ii) the reliance on iterative methods that are not guaranteed to converge to the correct solution.

This thesis sought to address the above issues through the application of rigorous numerical methods. The methods considered are interval analysis and deterministic global optimization. Hartree-Fock theory was used as the focal point of the studies performed.

Interval analysis techniques were applied to place rigorous bounds on numerical errors in Hartree-Fock calculations (Chapter 3). The significant contributions made included the development of methods for bounding truncation errors in the reduced incomplete gamma function, and for bounding errors propagated by the diagonalization of the Fock matrix. This led to the development of a program referred to as *Interval Hartree-Fock*. For an arbitrary closed-shell system, this program is able to compute rigorous error bounds on quantities including i) the total energy, ii) the molecular orbital energies, iii) the molecular orbital coefficients, and iv) derived electronic properties obtained from population analysis.

Interval Hartree-Fock can be used as a tool for worst-case error analysis. In Chapter 4, it was used to investigate the impact of various input and design related factors on numerical errors in Hartree-Fock computation. This included practical issues such as the accumulation of numerical errors due to increasing basis set and system size. The effect of less obvious, design related, factors were also considered.

These included i) the choice of interpolation scheme, ii) the variation of the integral screening threshold, iii) the use of both single and double precision arithmetic, and iv) the relaxation of error tolerances. The results point to potential issues with numerical errors when problem size increases; which can be allayed to a significant extent by applying accurate summation techniques. The results also highlight trade-offs in relation to arithmetic precision and fast memory usage that can be implemented without resulting in the loss of chemical accuracy. These can be applied, in particular, to work around the limitations of specialized processor hardware such as graphics processing units.

The thesis then develops an approach for solving the Hartree-Fock equations to within an arbitrarily specified margin of error (Chapter 5). This is achieved by treating problems in Hartree-Fock theory as problems in non-convex global optimization, which are solved by deterministic global optimization. The major contribution of this work is in the development of two approaches for generating linear relaxations of one and two-electron integrals. The first is based on decomposing each integral into a linear combination of $F_m(T)$ functions, while the second is based on a decomposition of each integral into a linear combination of generalized $(ss|ss)$-type integrals with two atomic centres. These approaches were implemented in COUENNE (*Convex Over and Under ENvelopes for Nonlinear Estimation*), an open source, general purpose, deterministic MINLP solver.

Computational results were presented for i) point energy calculation, ii) geometry optimization, iii) basis set optimization, and iv) excited state calculation (Chapter 6). In point energy calculations, the execution times for COUENNE was found to be comparable to those of BARON, and generally one to two orders of magnitude faster than GlobSol. It was also demonstrated that execution times can be reduced significantly by introducing symmetry-breaking constraints. The performance of COUENNE for geometry optimization was then analyzed. The results showed that performance varied significantly depending on the algorithm used for linear relaxation, bounds tightening, and branch selection. In larger systems, the fastest approaches were those that use the $(ss|ss)$ linear relaxation scheme combined with a branch selection algorithm based on strong branching. On the whole, the deterministic global optimization approach outlined in this thesis is much more costly than conventional methods such as SCF. However, it may be useful in problems where the global minimum is difficult to find, or when near-analytical guarantees on optimality are required. It can also be used to provide rigorous proofs for theorems that involve one and two-electron integral terms.

7.1 Future Work

The following sections highlight potential avenues for future work in the areas of interval analysis and deterministic global optimization.

7.1.1 Interval Analysis

This thesis demonstrated that *Interval Hartree-Fock* can be used effectively as a worst-case error analysis tool. This program is robust in that it can fluently model a diverse range of error related scenarios.

7.1.1.1 Further Error Analysis

The results presented in this thesis are confined to a limited set of small and medium-sized systems. Another next step is to consider larger systems involving a larger variety of atoms, molecules, and basis set types, and to explore further other factors that affect numerical accuracy such as near linearly dependent basis sets.

7.1.1.2 Addressing Performance Issues

The system size that can be considered is currently limited by the performance of *Interval Hartree-Fock*. This is due in part to the experimental nature of the code, the increased computational and memory requirements of interval computation, and the additional work required to guarantee containment. Extensions to the existing program should therefore work towards addressing these performance issues.

7.1.1.3 Application to Conventional SCF

The *Interval Hartree-Fock* program can be used as a tool for determining the source of numerical instability when convergence issues arise in a conventional SCF solver. This is achieved by performing the interval computation, inspecting the error bounds produced at each computational step, and identifying the source through a simple process of elimination. In preliminary experiments, *Interval Hartree-Fock* was able to detect numerical errors due to the presence of near-linearly dependent basis sets, and was able to identify the cause as being cancellation error during the Fock matrix construction step.

7.1.1.4 Interval SCF Iterations

The *Interval Hartree-Fock* program currently requires, as input, the set of M.O coefficients, from which error bounds on the total energy and other quantities are then derived. These bounds only reflect the numerical errors from a single SCF iteration, usually those of the final iteration that leads to convergence. However, by using the interval Fock matrix diagonalization approaches outlined in Section 3.5, it is also possible to analyze the numerical errors propagated across multiple SCF iterations, and even perform the entire SCF procedure via interval analysis.

An interval SCF method can potentially begin with an initial guess of $\mathbf{C_0} \in I(\mathbb{R}^{n\times n})$ that is assumed to bound the ground state molecular orbitals. At the first iteration, $\mathbf{C_0}$ is used to calculate an interval Fock matrix $\mathbf{F_0}$ which is then diagonalized to produce $\mathbf{C_1} \in I(\mathbb{R}^{n\times n})$. It can be shown that the intersection $\mathbf{C_1} = \mathbf{C_0} \bigcap \mathbf{C_1}$ provides a tighter bound on the ground state than $\mathbf{C_0}$. In principle, this procedure can be repeated to produce a sequence $\{\mathbf{C_0}, \mathbf{C_1}, \mathbf{C_2} .., \mathbf{C_n}\}$ that converges to the ground state.

The success of the above approach depends greatly on the ability to place tight bounds on the eigenvalues and eigenvectors of the interval Fock matrix. This has not been sufficiently demonstrated in this thesis. In future work, it may be worthwhile to consider the application of interval sub-division or branch and bound to the existing interval Fock matrix diagonalization code.

7.1.2 Deterministic Global Optimization

The Hartree-Fock extension to COUENNE developed in this thesis provides a robust software platform that can be built upon in future work. In its current form it can be used to solve model systems, and to provide rigorous computer assisted proofs for theorems. However, the results in Chapter 6 point to a number of practical computational difficulties that must first be addressed before it can be applied to larger systems.

7.1.2.1 Improved Convex Relaxations

The first task in future work should be to modify COUENNE and the underlying COIN-OR framework to accept general non-linear convex relaxations, in addition to linear relaxations. This will help facilitate a larger variety of relaxation types, such as αBB, semi-definite programs, geometric programming, etc.

The results in Chapter 6 show that $(ss|ss)$ relaxation is more efficient than

$F_m(T)$ relaxation for larger problems, because it aggregates together many $F_m(T)$ terms, thereby reducing the number of linear constraints generated during each convex relaxation step. This suggests that defining relaxations for larger compound expressions may be beneficial for solving larger systems. The software developed in this thesis can aid in the development of new relaxation approaches, as we have seen for example, by providing rigorous computer assisted proofs. The ideal, though perhaps unrealistic, goal is eventually to be able to generate the convex hull of the potential energy surface. The ongoing research in interpolated ab-initio potential energy surfaces may also be relevant to this effort.

7.1.2.2 Validated Linear Relaxation

The linear relaxation approaches developed in this thesis are not rigorous with respect to numerical errors. While they provide mathematically certain guarantees on optimality, assuming exact arithmetic, they may provide the wrong result in practice due to rounding error, truncation error, numerical instability etc. Interval analysis based global optimization approaches do not suffer from this problem, however as the results with `GlobSol` show, this usually comes at the cost of performance. In future work, it may be useful to consider developing linear relaxations for one and two electron integrals that take into account the effects of rounding errors. This could be based initially on the concept of *validated linear relaxations* proposed by Kearfott, Hongthong and others [68, 49, 69, 81].

7.1.2.3 Improving Problem Formulation

The formulation of the Hartree-Fock equations (Equation 5.2) considered in this thesis is relatively standard, in that it could be derived from any standard quantum chemistry textbook. Alternative formulations were not considered to any great extent. While the inclusion of symmetry-breaking constraints was modest in scope, it yielded substantial improvements to the execution time by eliminating symmetric solutions and reducing the size of the search space (Section 6.4.1). Alternative ab-initio quantum chemistry models were also not considered in this work. Fukuda and Zhao [35, 165] demonstrates promising results in solving the two-electron reduced density matrix (RDM) problem which is more amenable to a rigorous solution than the Hartree-Fock problem, since it is an instance of semi-definite programming. The positive results with respect to the symmetry-breaking constraints, and the RDM method both suggest that the mathematical programming formulation of the Hartree-Fock equations is an area that could be improved upon in future work.

Bibliography

[1] GAMS World Global Optimization Library. GAMS Development Corp. http://www.gamsworld.org/global/globallib.htm.

[2] *ILOG CPLEX 10.0 User's Manual*, January 2006.

[3] ACHTERBERG, T., KOCH, T., AND MARTIN, A. Branching rules revisited. *Oper. Res. Lett. 33*, 1 (2005), 42 – 54.

[4] ADJIMAN, C. S., ANDROULAKIS, I. P., AND FLOUDAS, C. A. A Global Optimization method, aBB for General Twice-Differentiable Constrained NLPs - I. Theoretical Advances, 1998.

[5] ADJIMAN, C. S., ANDROULAKIS, I. P., AND FLOUDAS, C. A. A Global Optimization method, aBB for General Twice-Differentiable Constrained NLPs - II Implementation and Computational Results, 1998.

[6] ALMLÖF, J. Direct methods in electronic structure theory. In *Modern Electronic Structure Theory Part II*, D. R. Yarkony, Ed., vol. 1. World Scientific, Singapore, 1995, pp. 110–151.

[7] ANDROULAKIS, M., MARANAS, C. D., ANDROULAKIS, I. P., AND FLOUDAS, C. A. A Deterministic Global Optimization Approach for the Protein Folding Problem. *J. Chem. Phys 100* (1996), 133–150.

[8] APPLEGATE, D., BIXBY, R., CHVÁTAL, V., AND COOK, W. On the Solution of Traveling Salesman Problems. In *International Congress of Mathematicians* (1998), pp. 645–656.

[9] APPLEGATE, D., BIXBY, R., CHVÁTAL, V., AND COOK, W. *The Traveling Salesman Problem, A Computational Study*. Princeton University Press, 2006.

[10] BARDO, R. D., AND RUEDENBERG, K. Even-tempered atomic orbitals. VI. Optimal orbital exponents and optimal contractions of Gaussian primitives for hydrogen, carbon, and oxygen in molecules. *J. Chem. Phys. 60*, 3 (1974), 918–931.

[11] BELOTTI, P., LEE, J., LIBERTI, L., MARGOT, F., AND WÄCHTER, A. Branching and bounds tightening techniques for non-convex MINLP. *Optim. Method Softw. 24*, 4-5 (2009), 597–634.

[12] BONAMI, P., BIEGLER, L. T., CONN, A. R., CORNUJOLS, G., GROSSMANN, I. E., LAIRD, C. D., LEE, J., LODI, A., MARGOT, F., SAWAYA, N., AND WCHTER, A. An algorithmic framework for convex mixed integer nonlinear programs. *Disc. Opt. 5*, 2 (2008), 186 – 204. In Memory of George B. Dantzig.

[13] BOUFERGUENE, A., FARES, M., AND HOGGAN, P. E. STOP: A Slater-type orbital package for molecular electronic structure determination. *Int. J. Quantum Chem. 57*, 4 (1996), 801–810.

[14] BRAUER, A. Limits for the characteristic roots of a matrix II. *Duke Math. J. 14* (1947), 21–26.

[15] BROOK, A., KENDRICK, D., AND MEERAUS, A. GAMS, a user's guide. *SIGNUM Newsl. 23* (December 1988), 10–11.

[16] BROYDEN, C. G. Quasi-Newton methods and their application to function minimization. *Math. Comput. 21* (1967), 368–381.

[17] CAFIERI, S., LEE, J., AND LIBERTI, L. On convex relaxations of quadrilinear terms. *J. Global Optim. 47* (August 2010), 661–685.

[18] CAPRANI, O., MADSEN, K., AND NIELSEN, H. B. Introduction to interval analysis, Nov 2002.

[19] CRAWFORD, J. M., GINSBERG, M. L., LUKS, E. M., AND ROY, A. Symmetry-Breaking Predicates for Search Problems. In *Principles of Knowledge Representation and Reasoning* (1996), pp. 148–159.

[20] DAVIDSON, E. R., AND FELLER, D. Basis set selection for molecular calculations. *Chem. Rev. 86*, 4 (1986), 681–696.

[21] DEKKER, T. J. A floating-point technique for extending the available precision. *Numerische Mathematik 18* (1971), 224–242. 10.1007/BF01397083.

[22] ESPOSITO, W. R., AND FLOUDAS, C. A. Deterministic global optimization in isothermal reactor network synthesis. *J. Global Optim. 22* (January 2002), 59–95.

[23] FAEGRI, K., AND ALMLOF, J. Energy-optimized GTO basis sets for LCAO Calculations. A Gradient Approach. *J. Comput. Chem. 7*, 4 (1986), 396–405.

[24] FALK, J., AND SOLAND, R. An algorithm for separable nonconvex programming. *Management Sci. 15* (1969), 550–569.

[25] FLETCHER, R. *Practical methods of optimization; (2nd ed.).* Wiley-Interscience, New York, NY, USA, 1987.

[26] FLOUDAS, C. A. *Deterministic Global Optimization: Theory, Methods and Applications*. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2005.

[27] FLOUDAS, C. A., AND GOUNARIS, C. E. A review of recent advances in global optimization. *J. Global Optim. 45* (September 2009), 3–38.

[28] FLOUDAS, C. A., AND VISWESWARAN, V. A global optimization algorithm (GOP) for certain classes of nonconvex NLPs–I. Theory. *Comput. Chem. Eng. 14*, 12 (1990), 1397 – 1417.

[29] FORESMAN, J., AND FRISCH, A. *Exploring Chemistry with Electronic Structure Methods*. Gaussian, Inc., New Haven, CT, 1996.

[30] FORREST, J., , AND LOUGEE-HEIMER, R. *CBC User Guide*. IBM, 2005. (Accessed Feb 7, 2011).

[31] FORREST, J., DE LA NUEZ, D., AND LOUGEE-HEIMER, R. *CLP User Guide*. IBM, 2004. (Accessed Feb 7, 2011).

[32] FOURER, R., GAY, D. M., AND KERNIGHAN, B. Algorithms and model formulations in mathematical programming. Springer-Verlag New York, Inc., New York, NY, USA, 1989, ch. AMPL: a mathematical programming language, pp. 150–151.

[33] FRANK, W. L. Computing Eigenvalues of Complex Matrices by Determinant Evaluation and by Methods of Danilewski and Wielandt. *Journal of the Society for Industrial and Applied Mathematics 6*, 4 (1958), pp. 378–392.

[34] FUCHS, M., AND NEUMAIER, A. Handling Uncertainty in Higher Dimensions with Potential Clouds towards Robust Design Optimization. In *Soft Methods for Handling Variability and Imprecision*, D. Dubois, M. Lubiano, H. Prade, M. Gil, P. Grzegorzewski, and O. Hryniewicz, Eds., vol. 48 of *Advances in Soft Computing*. Springer Berlin / Heidelberg, 2008, pp. 376–382.

[35] FUKUDA, M., BRAAMS, B. J., NAKATA, M., OVERTON, M. L., PERCUS, J. K., YAMASHITA, M., AND ZHAO, Z. Large-scale semidefinite programs in electronic structure calculation. *Math. Program. 109*, 2-3 (2007), 553–580.

[36] GERSCHGORIN, S. Über die Abgrenzung der Eigenwerte einer Matrix. *Bulletin de l'Académie des Sciences de l'URSS. Classe des sciences mathématiques et na* (1931), 749–754.

[37] GILBERT, A. T. B., BESLEY, N. A., AND GILL, P. M. W. Self-Consistent Field Calculations of Excited States Using the Maximum Overlap Method (MOM). *J. Phys. Chem. A 112*, 50 (2008), 13164–13171. PMID: 18729344.

[38] GILL, P. M., JOHNSON, B. G., AND POPLE, J. A. Two-Electron Repulsion Integrals Over Gaussian s Functions. *Int. J. Quantum Chem. 40* (1991), 745–752.

[39] GILL, P. M. W. Molecular integrals over gaussian basis functions. *Adv. Quantum Chem. 25* (1994), 141–205.

[40] GILL, P. M. W., HEAD-GORDON, M., AND POPLE, J. A. Efficient Computation of Two-Electron-Repulsion Integrals and Their nth-Order Derivatives Using Contracted Gaussian Basis Sets. *J. Phys. Chem. 94* (1990), 5564–5572.

[41] HANSEN, E. A globally convergent interval method for computing and bounding real roots. *BIT Numerical Mathematics 18* (1978), 415–424. 10.1007/BF01932020.

[42] HANSEN, E. *Global optimization using interval analysis*. Marcel Dekker, Inc., New York, NY, 1992.

[43] HANSEN, E., AND WALSTER, G. *Global optimization using interval analysis*, second ed. Marcel Dekker, Inc., New York, NY, 2004.

[44] HEAD-GORDON, M., AND POPLE, J. A. A method for two-electron Gaussian integral and integral derivative evaluation using recurrence relations. *J. Comput. Phys. 26* (1978), 218–231.

[45] HEATH, M. T. *Scientific Computing: An Introductory Survey*. McGraw-Hill Higher Education, 1996.

[46] HIGHAM, N. J. *Accuracy and Stability of Numerical Algorithms*, second ed. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 2002.

[47] HLADÍK, M., DANEY, D., AND TSIGARIDAS, E. Bounds on Real Eigenvalues and Singular Values of Interval Matrices. *SIAM J. Matrix Anal. Appl. 31*, 4 (2010), 2116–2129.

[48] HOLLAND, J. H. Genetic Algorithms and the Optimal Allocation of Trials. *SIAM J. Comput. 2*, 2 (1973), 88–105.

[49] HONGTHONG, S., AND KEARFOTT, R. Rigorous Linear Overestimators and Underestimators. Tech. rep., University of Louisiana, 2004.

[50] HORST, R., AND PARDALOS, P. *Handbook of Global Optimization*. Kluwer, Dordrecht, 1995.

[51] HUZINAGA, S. Gaussian-Type Functions for Polyatomic Systems. I. *J. Chem. Phys. 42*, 4 (1965), 1293–1302.

[52] HUZINAGA, S., AND ARNAU, C. Virtual Orbitals in Hartree-Fock Theory. *Phys. Rev. A 1*, 5 (May 1970), 1285–1288.

[53] HUZINAGA, S., AND ARNAU, C. Virtual Orbitals in Hartree–Fock Theory. II. *J. Chem. Phys. 54*, 5 (1971), 1948–1951.

[54] HUZINAGA, S., AND KLOBUKOWSKI, M. Well-tempered Gaussian basis sets for the calculation of matrix Hartree–Fock wavefunctions. *Chem. Phys. Lett. 212*, 3-4 (1993), 260 – 264.

[55] IEEE COMPUTER SOCIETY STANDARDS COMMITTEE. WORKING GROUP OF THE MICROPROCESSOR STANDARDS SUBCOMMITTEE, AND AMERICAN NATIONAL STANDARDS INSTITUTE. *IEEE standard for binary floating-point arithmetic*. ANSI/IEEE Std 754-1985. IEEE Computer Society Press, 1109 Spring Street, Suite 300, Silver Spring, MD 20910, USA, 1985.

[56] JANES, P. P. http://cs.anu.edu.au/people/Pete.Janes/applications/ (accessed Jan 11, 2011).

[57] JANES, P. P., AND RENDELL, A. P. Deterministic global optimization in ab-initio quantum chemistry. *J. Global Optim.*, 1–22. 10.1007/s10898-012-9868-5.

[58] JANES, P. P., AND RENDELL, A. P. Including Rigorous Numerical Bounds in Quantum Chemistry Calculations: Gaussian Integral Evaluation. *CSE 0* (2008), 75–82.

[59] JANES, P. P., AND RENDELL, A. P. Placing Rigorous Bounds on Numerical Errors in HartreeFock Energy Computations. *J. Chem. Theory Comput. 7*, 6 (2011), 1631–1639.

[60] JANSSON, C. Interval linear systems with symmetric matrices, skew-symmetric matrices and dependencies in the right hand side. *Computing 46* (1991), 265–274. 10.1007/BF02238302.

[61] KAHAN, W. Pracniques: further remarks on reducing truncation errors. *Commun. ACM 8* (January 1965), 40–.

[62] KAHLE, J. A., DAY, M. N., HOFSTEE, H. P., JOHNS, C. R., MAEURER, T. R., AND SHIPPY, D. Introduction to the cell multiprocessor. *IBM J. Res. Dev. 49* (July 2005), 589–604.

[63] KARUSH, W. Minima of Functions of Several Variables with Inequalities as Side Constraints. Master's thesis, Department of Mathematics, University of Chicago, 1939.

[64] KEARFOTT, B. GlobSol. http://interval.louisiana.edu/GlobSol/download_GlobSol.html.

[65] KEARFOTT, B. R. Preconditioners for the Interval Gauss-Seidel Method. *SIAM J. Numer. Anal. 27*, 3 (1990), pp. 804–822.

[66] KEARFOTT, B. R. *Rigorous Global Search: Continuous Problems*. Kluwer, Dordrecht, 1996.

[67] KEARFOTT, B. R. GlobSol user guide. *Optim. Method Softw. 24* (August 2009), 687–708.

[68] KEARFOTT, R. B. Discussion and Empirical Comparisons of Linear Relaxations and Alternate Techniques in Validated Deterministic Global Optimization, 2004.

[69] KEARFOTT, R. B., AND HONGTHONG, S. Validated Linear Relaxations and Preprocessing: Some Experiments. *SIAM J. Optimiz. 16*, 2 (2005), 418–433.

[70] KENNEDY, J., AND EBERHART, R. Particle Swarm Optimization. In *Proceedings of IEEE International Conference on Neural Networks* (1995), vol. IV, pp. 1942–1948.

[71] KIRKPATRICK, S., GELATT, C. D., AND VECCHI, M. P. Optimization by Simulated Annealing. *Science 220*, 4598 (1983), 671–680.

[72] KNIZIA, G., LI, W., SIMON, S., AND WERNER, H.-J. Determining the numerical stability of quantum chemistry algorithms. *J. Chem. Theory Comput. 7*, 8 (2011), 2387–2398.

[73] KNUTH, D. *The Art of Computer Programming: Seminumerical Algorithms*, vol. 2. Addison-Wesley, Reading, Massachusetts, 1969.

[74] KRAWCZYK, R. Newton-Algorithmen zur Bestimmung von Nullstellen mit Fehlerschranken. *Computing (Arch. Elektron. Rechnen) 4* (1969).

[75] KUCHERENKO, S., AND SYTSKO, Y. Application of Deterministic Low-Discrepancy Sequences in Global Optimization. *Comput. Optim. Appl. 30* (2005), 297–318. 10.1007/s10589-005-4615-1.

[76] KUHN, G., AND TUCKER, A. Nonlinear Programming. In *Proceeding of the 2nd Berkeley Symposium on Mathematical Statistics and Probability* (1951), J. Neyman, Ed., University of California Press, pp. 481–493.

[77] LAND, A. H., AND DOIG, A. G. An Automatic Method of Solving Discrete Programming Problems. *Econometrica 28*, 3 (1960), 497–520.

[78] LAVOR, C. A deterministic approach for global minimization of molecular potential energy functions. *Int. J. Quantum Chem. 95*, 3 (2003), 336–343.

[79] LAVOR, C. C., CARDOZO, T. M., AND NASCIMENTO, M. A. C. Using an interval branch-and-bound algorithm in the Hartree-Fock method. *Int. J. Quantum Chem. 103*, 5 (2005), 500–504.

[80] LAVOR, C. , LIBERTI, L. , MACULAN, N. , AND NASCIMENTO, M. A. C. Solving Hartree-Fock systems with global optimization methods. *Europhys. Lett. 77*, 5 (Mar 2007), 50006.

[81] LEBBAH, Y., MICHEL, C., AND RUEHER, M. An efficient and safe framework for solving optimization problems. *J. Comput. Appl. Math. 199*, 2 (2007), 372 – 377. Special Issue on Scientific Computing, Computer Arithmetic, and Validated Numerics (SCAN 2004), Special Issue on Scientific Computing, Computer Arithmetic, and Validated Numerics (SCAN 2004).

[82] LIBERTI, L. Ev3: A Library for Symbolic Computation in C++ using n-ary Trees LIX,. Tech. rep., LIX, Ecole Polytechnique 91128 Palaiseau, France, 2003.

[83] LIBERTI, L. *Reformulation and Convex Relaxation Techniques for Global Optimization*. PhD thesis, Imperial College London, March 2004.

[84] LIBERTI, L. *Introduction to Global Optimization*, February 2008. http://www.lix.polytechnique.fr/~liberti/teaching/globalopt-lima.pdf (Accessed 1 Jun 2011).

[85] LIBERTI, L. Reformulations in mathematical programming: automatic symmetry detection and exploitation. *Math. Prog.* (2010), 1–32. 10.1007/s10107-010-0351-0.

[86] LIBERTI, L., LAVOR, C., MACULAN, N., AND NASCIMENTO, M. A. C. Reformulation in mathematical programming: An application to quantum chemistry. *Discrete Appl. Math. 157* (March 2009), 1309–1318.

[87] LIBERTI, L., AND PANTELIDES, C. C. Convex Envelopes of Monomials of Odd Degree. *J. Global Optim. 25* (2003), 157–168. 10.1023/A:1021924706467.

[88] LIN, Y. *LP-Based Strategies for Deterministic Global Optimization using Interval Methods*. PhD thesis, University of Notre Dame, July 2004.

[89] LIN, Y., AND STADTHERR, M. A. Deterministic global optimization of molecular structures using interval analysis. *J.Comput. Chem. 26*, 13 (2005), 1413–1420.

[90] LITTLE, J., MURTY, K., SWEENEY, D., AND KAREL, C. An algorithm for the travelling salesman problem. *Oper. Res. 11* (1963), 972–989.

[91] LOH, E., AND WALSTER, G. W. Rump's Example Revisited. *Reliable Computing 8* (2002), 245–248. 10.1023/A:1015569431383.

[92] LOUGEE-HEIMER, R. The Common Optimization INterface for Operations Research: Promoting open-source software in the operations research community. *IBM J. Res. Dev. 47* (January 2003), 57–66.

[93] LUEHR, N., UFIMTSEV, I. S., AND MARTINEZ, T. J. Dynamic precision for electron repulsion integral evaluation on graphical processing units (gpus). *J. Chem. Theory Comput. 7*, 4 (2011), 949–954.

[94] MADSEN, K., NIELSEN, H., AND TINGLEFF, O. Optimization with Constraints, March 2004.

[95] MAKINO, K., AND BERZ, M. Remainder Differential Algebras and their Applications. In *Computational Differentiation: Techniques, Applications, and Tools*, M. Berz, C. Bischof, G. Corliss, and A. Griewank, Eds. SIAM, Philadelphia, PA, 1996, pp. 63–74.

[96] MARANAS, C., MCDONALD, C., HARDING, S., AND FLOUDAS, C. Locating all azeotropes in homogeneous azeotropic systems. *Comput. Chem. Eng. 20*, Supplement 1 (1996), S413 – S418. European Symposium on Computer Aided Process Engineering-6.

[97] MARANAS, C. D., AND FLOUDAS, C. A. A global optimization approach for Lennard-Jones microclusters. *J. Chem. Phys. 97*, 10 (1992), 7667–7678.

[98] MARANAS, C. D., AND FLOUDAS, C. A. Finding all solutions of nonlinearly constrained systems of equations. *J. Global Optim. 7* (1995), 143–182. 10.1007/BF01097059.

[99] MARUYAMA, T., YOSHIDA, T., KAN, R., YAMAZAKI, I., YAMAMURA, S., TAKAHASHI, N., HONDOU, M., AND OKANO, H. Sparc64 VIIIfx: A New-Generation Octocore Processor for Petascale Computing. *IEEE Micro 30* (2010), 30–40.

[100] MCCORMICK, G. Converting general nonlinear programming problems to separable nonlinear programming problems. Tech. Rep. T-267, George Washington University, Washington, DC, 1972.

[101] MCCORMICK, G. P. Computability of global solutions to factorable nonconvex programs: Part I Convex underestimating problems. *Math. Program. 10* (1976), 147–175. 10.1007/BF01580665.

[102] McDonald, C. M., and Floudas, C. A. Decomposition based and branch and bound global optimization approaches for the phase equilibrium problem. *J. Global Optim. 5* (1994), 205–251. 10.1007/BF01096454.

[103] McMurchie, L. E., and Davidson, E. R. One- and Two-Electron Integrals over Cartesian Guassian Functions. *J. Chem. Phys. 89*, 9 (1988), 5777–5786.

[104] Messine, F. Deterministic global optimization using interval constraint propagation techniques. *Rairo-Rech Oper 38* (2004), 277–293.

[105] Milthorpe, J. Using Interval Analysis to Bound Numerical Errors in Scientific Computing. Honours Thesis, Department of Computer Science, Australian National University, October 2005.

[106] Mladenovic, N., Petrovic, J., Kovacevic-Vujcic, V., and Cangalovic, M. Solving spread spectrum radar polyphase code design problem by tabu search and variable neighbourhood search. *Eur. J. Oper. Res. 151*, 2 (2003), 389 – 399. Meta-heuristics in combinatorial optimization.

[107] Moore, R., and Yang, C. Interval analysis I. Tech. Rep. LMSD-285875, Lockheed Missiles and Space Company, Sunnyvale, CA, USA, 1959.

[108] Moore, R. E. Automatic Error Analysis in Digital Computation. Tech. Rep. LMSD-48421, Lockheed Missiles and Space Company, Sunnyvale, CA, USA, Jan 1959.

[109] Moore, R. E., and Jones, S. T. Safe Starting Regions for Iterative Methods. *SIAM J. Numer. Anal. 14*, 6 (1977), 1051–1065.

[110] Neumaier, A. On Shary's Algebraic Approach for Linear Interval Equations. *SIAM J. Matrix Anal. Appl. 21* (March 2000), 1156–1162.

[111] Neumaier, A. Complete search in continuous global optimization and constraint satisfaction. *Acta Numerica 13* (2004), 271–369.

[112] Neumaier, A., Shcherbina, O., Huyer, W., and Vink, T. A comparison of complete global optimization solvers. *Math. Program. 103* (2005), 335–356. 10.1007/s10107-005-0585-4.

[113] Neumeier, A. *Interval methods for systems of equations.* Cambridge University Press, 1990.

[114] NVIDIA. NVIDIA CUDA Programming Guide Version 3.0, February 2010.

[115] OBARA, S., AND SAIKA, A. Efficient recursive computation of molecular integrals over Cartesian Gaussian functions . *J Chem. Phys. 84*, 7 (1986), 3963.

[116] OGITA, T., RUMP, S. M., AND OISHI, S. Accurate Sum and Dot Product. *SIAM J. Sci. Comput. 26*, 6 (2005), 1955–1988.

[117] POWELL, M. A method for nonlinear constraints in minimization problems. In *Optimization* (London, 1969), R. Fletcher and A. Harwell, Eds., Academic Press, pp. 283–298.

[118] PRIEST, D. *On Properties of Floating Point Arithmetics: Numerical Stability and the Cost of Accurate Computations*. PhD thesis, Mathematics Department, University of California, Berkeley, CA, USA, November 2002.

[119] PULAY, P. Convergence acceleration of iterative sequences - The case of SCF iteration. *Chem. Phys. Lett. 73*, 2 (1980), 393 – 398.

[120] RAMDAS, T., EGAN, G., ABRAMSON, D., AND BALDRIDGE, K. Towards a special-purpose computer for HartreeFock computations. *Theor. Chem. Acc. 120* (2008), 133–153.

[121] RENDELL, A., CLARKE, B., JANES, P., MILTHORPE, J., AND YANG, R. Interval arithmetic and computational science: Rounding and truncation errors in n-body methods. In *Computational Science and its Applications, 2007. ICCSA 2007. International Conference on* (aug. 2007), pp. 457 –466.

[122] RENDELL, A. P., AND CLARKE, B. Techniques for accurate interval and floating point summation. to be published.

[123] ROHN, J. A Handbook of Results on Interval Linear Problems. Tech. rep., Czech Academy of Sciences, Prague, Czech Republic, April 2005.

[124] ROLAND LINDH. Integrals of Electron Repulsion. In *Encyclopedia of Computational Chemistry* (1998), P. v. R. Schleyer et. al, Ed., vol. 2, Wiley, p. 1337.

[125] RUEDENBERG, K., RAFFENETTI, R., AND BARDO, R. Energy, Structure and Reactivity. In *Proceedings of the 1972 Boulder Conference* (New York, 1973), Wiley.

[126] RUMP, S. *Kleine Fehlerschranken bei Matrixproblemen*. PhD thesis, University of Karlsruhe, 1980.

[127] RUMP, S. Algorithms for Verified Inclusions - Theory and Practice. In *Reliability in Computing: The Role of Interval Methods in Scientific Computing* (1988), R. Moore, Ed., vol. 19, Academic Press, pp. 109–126.

[128] RYS, J., DUPUIS, M., AND KING, H. Computation of electron repulsion integrals using the rys quadrature method. *J. Comput. Chem. 4* (1983), 154–157.

[129] SAHINIDIS, N. V. *BARON Branch And Reduce Optimization Navigator User's Manual Version 4.0.* Univeristy of Illinois at Urbana-Champaign, 2000.

[130] SAHNI, S. Computationally Related Problems. *SIAM J. Comput. 3*, 4 (1974), 262–279.

[131] SALTELLI, A. Sensitivity analysis for importance assessment. *Risk Analysis 22*, 3 (2002), 579–590.

[132] SALTELLI, A. Global Sensitivity Analysis: An Introduction. In *The Fourth International Conference on Sensitivity Analysis of Model Output - SAMO 2004* (2004), Kenneth M. Hanson and Franois M. Hemez, Ed., pp. 27–43.

[133] SALTELLI, A., CHAN, K., AND SCOTT, E., Eds. *Sensitivity Analysis.* John Wiley and Sons, Ltd, New York, 2001.

[134] SALTELLI, A., RATTO, M., TARANTOLA, S., AND CAMPOLONGO, F. Sensitivity analysis practices: Strategies for model-based inference. *RESS 91*, 10-11 (2006), 1109 – 1125. The Fourth International Conference on Sensitivity Analysis of Model Output - SAMO 2004.

[135] SCHLEGEL, H. B. Exploring potential energy surfaces for chemical reactions: An overview of some practical methods. *Journal of Computational Chemistry 24*, 12 (2003), 1514–1527.

[136] SCHMIDT, M. W., BALDRIDGE, K. K., BOATZ, J. A., ELBERT, S. T., GORDON, M. S., JENSEN, J. H., KOSEKI, S., MATSUNAGA, N., NGUYEN, K. A., SU, S., WINDUS, T. L., DUPUIS, M., AND MONTGOMERY, JR., J. A. General atomic and molecular electronic structure system. *J. Comput. Chem. 14* (November 1993), 1347–1363.

[137] SCHUCHARDT, K. L., DIDIER, B. T., ELSETHAGEN, T., SUN, L., GURUMOORTHI, V., CHASE, J., LI, J., AND WINDUS, T. L. Basis Set Exchange: A Community Database for Computational Sciences. *Journal of Chemical Information and Modeling 47*, 3 (2007), 1045–1052. PMID: 17428029.

[138] SHARY, S. P. Algebraic approach in the "outer problem" for interval linear equations. *Reliable Computing 3*, 2 (1997), 103–135.

[139] SHAVITT, I. The Gaussian Function in Calculations of Statistical Mechanics and Quantum Mechanics. *Meth. Comput. Phys. 2* (1963), 1–44.

[140] SHERALI, H. D., AND ADAMS, W. P. *A reformulation-linearization technique for solving discrete and continuous nonconvex problems*. Kluwer Academic, Dordrecht, 1999.

[141] SHERALI, H. D., AND ALAMEDDINE, A. A new reformulation-linearization technique for bilinear programming problems. *J. Global Optim. 2* (1992), 379–410. 10.1007/BF00122429.

[142] SMITH, E. *On the Optimal Design of Continuous Processes*. PhD thesis, Imperial College of Science Technology and Medicine University of London, October 1996.

[143] SUN MICROSYSTEMS INC. The Sun Fire V1280 Server Architecture, November 2002. (Accessed Jan 9, 2011).

[144] SUN MICROSYSTEMS INC. Sun Studio 11: C++ Interval Arithmetic Programming Reference, 2005. (Accessed Feb 6, 2011).

[145] SUN MICROSYSTEMS INC. Sun Studio 11: Sun Performance Library User's Guide, November 2005. (Accessed Aug 17, 2011).

[146] SUN MICROSYSTEMS INC. SunStudio 11: C++ User's Guide, 2005. (Accessed Feb 6, 2011).

[147] SWILLIAM J. HUNT, AND III, W. A. G. Excited States of H2O using improved virtual orbitals. *Chem. Phys. Lett. 3*, 6 (1969), 414 – 418.

[148] SZABO, A., AND OSTLUND, N. S. *Modern Quantum Chemistry: Introduction to Advanced Electronic Structure Theory*. Dover Publications, July 1996.

[149] TAKASHIMA, H., AMISAKI, T., KITAMURA, K., AND NAGASHIMA, U. Numerical accuracy on $F_m(z)$ for molecular integral calculations. *Comput. Phys. Commun. 148* (2002), 182–187.

[150] TAKASHIMA, H., KITAMURA, K., TANABE, K., AND NAGASHIMA, U. Is large-scale ab initio Hartree-Fock calculation chemically accurate? Toward improved calculation of biological molecule properties. *J. Comput. Chem. 20*, 4 (1999), 443–454.

[151] TAWARMALANI, M., AND SAHINIDIS, N. V. *Convexification and Global Optimization in Continuous And Mixed-Integer Nonlinear Programming*. Kluwer Academic Publishers, Norwell, MA, USA, 2002.

[152] UFIMTSEV, I. S., AND MARTINEZ, T. J. Quantum Chemistry on Graphical Processing Units. 1. Strategies for Two-Electron Integral Evaluation. *J. Chem. Theory Comput. 4*, 2 (2008), 222–231.

[153] UFIMTSEV, I. S., AND MARTINEZ, T. J. Quantum Chemistry on Graphical Processing Units. 2. Direct Self-Consistent-Field Implementation. *J. Chem. Theory Comput. 5*, 4 (2009), 1004–1015.

[154] UFIMTSEV, I. S., AND MARTINEZ, T. J. Quantum Chemistry on Graphical Processing Units. 3. Analytical Energy Gradients, Geometry Optimization, and First Principles Molecular Dynamics. *J. Chem. Theory Comput. 5*, 10 (2009), 2619–2628.

[155] VISWESWARAN, V., AND FLOUDAS, C. A. New properties and computational improvement of the GOP algorithm for problems with quadratic objective functions and constraints. *J. Global Optim. 3* (1993), 439–462. 10.1007/BF01096414.

[156] WAECHTER, A., AND BIEGLER, L. T. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program. 106* (May 2006), 25–57.

[157] WALES, D., DOYLE, J., DULLWEBER, A., HODGES, M., NAUMKIN, F., CALVO, F., HERNÁNDEZ-ROJAS, J., AND MIDDLETON, T. The Cambridge Cluster Database. (Accessed Jan 2, 2011).

[158] WALES, D. J., AND DOYE, J. P. K. Global Optimization by Basin-Hopping and the Lowest Energy Structures of Lennard-Jones Clusters Containing up to 110 Atoms. *J. Phys. Chem. A 101*, 28 (1997), 5111–5116.

[159] WALSH, T. General Symmetry Breaking Constraints. In *Principles and Practice of Constraint Programming - CP 2006*, F. Benhamou, Ed., vol. 4204 of *Lecture Notes in Computer Science*. Springer Berlin / Heidelberg, 2006, pp. 650–664. 10.1007/11889205_46.

[160] WENZEL, W., AND HAMACHER, K. Stochastic Tunneling Approach for Global Minimization of Complex Potential Energy Landscapes. *Phys. Rev. Lett. 82*, 15 (Apr 1999), 3003–3007.

[161] WOLFE, J. M. Reducing truncation errors by programming. *Commun. ACM 7* (June 1964), 355–356.

[162] YASUDA, K. Two-electron integral evaluation on the graphics processor unit. *J. Comput. Chem. 29* (2008), 334–342.

[163] ZASLAVSKI, A., AND LIBERTI, L. Writing Global Optimization Software. In *Global Optimization*, P. Pardalos, L. Liberti, and N. Maculan, Eds., vol. 84 of *Nonconvex Optimization and Its Applications*. Springer US, 2006, pp. 211–262. 10.1007/0-387-30528-9_8.

[164] ZEIN, A., MCCREATH, E., RENDELL, A., AND SMOLA, A. Performance Evaluation of the NVIDIA GeForce 8800 GTX GPU for Machine Learning. In *Proceedings of the 8th international conference on Computational Science, Part I* (Berlin, Heidelberg, 2008), ICCS '08, Springer-Verlag, pp. 466–475.

[165] ZHAO, Z., BRAAMS, B. J., FUKUDA, M., OVERTON, M. L., AND PERCUS, J. K. The reduced density matrix method for electronic structure calculations and the role of three-index representability conditions. *J. Chem. Phys. 120*, 5 (2004), 2095–2104.

[166] ZORN, K., AND SAHINIDIS, N. Hartree Fock Self-Consistent Calculations: Global Optimization of Electronic Structure. In *Advances in Optimization II* (2010).

Author Index